

500.40174X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE



Applicant(s): KASHIWAGI, Yugo
Serial No.: (Not yet assigned)
Filed: May 31, 2001
Title: SYSTEM DEVELOPING METHOD, STORAGE MEDIUM,
INFORMATION PROCESSING APPARATUS, INFORMATION
TERMINAL APPARATUS, INFORMATION PROCESSING
SYSTEM, AND INFORMATION PROCESSING METHOD

LETTER CLAIMING RIGHT OF PRIORITY

Honorable Commissioner of
Patents and Trademarks
Washington, D.C. 20231

May 31, 2001

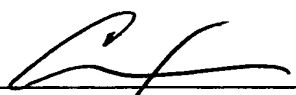
Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the
applicant(s) hereby claim(s) the right of priority based on Japanese
Patent Application No.(s) 2000-166468, filed May 31, 2000.

A certified copy of said Japanese Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP



Carl I. Brundidge
Registration No. 29,621

CIB/alb
Attachment
(703)312-6600

日 本 国 特 許 庁

PATENT OFFICE
JAPANESE GOVERNMENT

JC978 U.S. PTO
09/867615
05/31/01

別紙添付の書類に記載されている事項は下記の出願書類に記載されて
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed
with this Office.

出 願 年 月 日

Date of Application:

2000年 5月31日

願 番 号

Application Number:

特願2000-166468

願 人

Applicant (s):

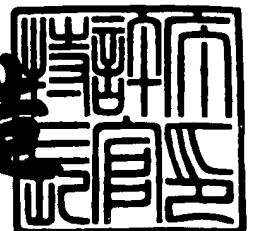
株式会社日立製作所

CERTIFIED COPY OF
PRIORITY DOCUMENT

2001年 4月13日

特許庁長官
Commissioner,
Patent Office

及川耕造



【書類名】 特許願

【整理番号】 H00002961

【提出日】 平成12年 5月31日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/44

【発明者】

【住所又は居所】 東京都小平市上水本町五丁目 2 0 番 1 号 株式会社日立
製作所 半導体グループ内

【氏名】 柏木 有吾

【特許出願人】

【識別番号】 000005108

【氏名又は名称】 株式会社日立製作所

【代理人】

【識別番号】 100089071

【弁理士】

【氏名又は名称】 玉村 静世

【電話番号】 047-361-8861

【手数料の表示】

【予納台帳番号】 011040

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 システム開発方法、記憶媒体、情報処理装置、情報端末装置、
情報処理システム及び情報処理方法

【特許請求の範囲】

【請求項 1】 複数の記述を有するソースプログラムにおける第 1 の記述を、
第 1 の計算機で実行可能な第 1 コードに変換し、

前記第 1 の記述を、該第 1 の記述によって定義される機能とは異なる機能を表
すところのコードであって、第 2 の計算機で実行可能な第 2 コードに変換すると
共に、前記ソースプログラムにおける第 2 の記述を、前記第 2 の計算機で実行可
能なその他のコードに変換し、

前記ソースプログラムのデバッグにおいて前記第 2 の計算機による前記第 2 コ
ードの実行に応答して前記第 1 の計算機が前記第 1 コードを実行することを特徴
とするシステム開発方法。

【請求項 2】 前記第 1 の計算機は、前記第 1 コードを実行する前に前記第
2 計算機から情報を取り込むことを特徴とする請求項 1 記載のシステム開発方法
。

【請求項 3】 前記第 2 コードは割り込み命令を表すコードであり、該コー
ドの実行により前記第 1 の計算機へ、前記第 2 コードのアドレス情報を供給する
ことを特徴とする請求項 2 記載のシステム開発方法。

【請求項 4】 前記第 2 コードのアドレス情報と前記第 1 コードとを対応さ
せる対照データを第 1 の計算機が参照可能に生成することを特徴とする請求項 1
記載のシステム開発方法。

【請求項 5】 コンパイラを用いて前記第 1 コードへの変換、第 2 コードへ
の変換、及びその他のコードへの変換を行うことを特徴とする請求項 1 乃至 4 の
何れか 1 項記載のシステム開発方法。

【請求項 6】 デバッガを用いて前記ソースプログラムのデバッグを行なう
ことを特徴とする請求項 5 記載のシステム開発方法。

【請求項 7】 前記第 1 の計算機はデバッガシステムを構成する計算機であ
り、前記第 2 の計算機はターゲット装置を構成する計算機であることを特徴とす

る請求項 1 乃至 4 の何れか 1 項記載のシステム開発方法。

【請求項 8】 前記第 1 の計算機はホスト装置を構成する計算機であり、前記第 2 の計算機はターゲット装置である端末装置を構成する計算機であることを特徴とする請求項 1 乃至 4 の何れか 1 項記載のシステム開発方法。

【請求項 9】 第 1 の記述と第 2 の記述を含むソースプログラムを入力する入力ステップと、入力した第 1 の記述を第 1 の計算機によって実行可能なコードに変換する第 1 変換ステップと、入力した前記第 1 の記述を該第 1 の記述により表される機能とは異なる機能であって、第 2 の計算機によって実行可能なコードへ変換する第 2 変換ステップと、入力した第 2 の記述を前記第 2 の計算機によって実行可能なコードに変換する第 3 変換ステップとを実行するプログラムを計算機によって読み取り可能に記憶して成るものであることを特徴とする記憶媒体。

【請求項 10】 前記プログラムは、1 個の記憶媒体に格納されていることを特徴とする請求項 9 記載の記憶媒体。

【請求項 11】 前記プログラムは、伝送線を介して記憶媒体に取り込まれることを特徴とする請求項 9 記載の記憶媒体。

【請求項 12】 前記第 2 の計算機によって実行可能なコードは、割り込みを表す割り込みコードであることを特徴とする請求項 9 乃至 11 の何れか 1 項記載の記憶媒体。

【請求項 13】 前記プログラムは更に、前記割り込みコードにリンクされる割り込みハンドラを規定するステップを含むことを特徴とする請求項 12 記載の記憶媒体。

【請求項 14】 前記割り込みコードは、その実行により、前記第 2 の計算機のオペレーティングシステムが規定する割り込みハンドラの処理に分岐するものであることを特徴とする請求項 12 記載の記憶媒体。

【請求項 15】 前記プログラムは更に、前記第 2 の計算機における前記割り込みコードのアドレスと前記第 1 の計算機が実行可能なコードとの対応を示すテーブルを形成するステップを含むことを特徴とする請求項 12 記載の記憶媒体。

【請求項 1 6】 前記第 1 の計算機はデバッグシステムを構成する計算機であり、前記第 2 の計算機はターゲット装置を構成する計算機であることを特徴とする請求項 9 乃至 1 7 の何れか 1 項記載の記憶媒体。

【請求項 1 7】 前記第 1 の計算機はホスト装置を構成する計算機であり、前記第 2 の計算機は情報端末装置を構成する計算機であることを特徴とする請求項 9 乃至 1 5 の何れか 1 項記載の記憶媒体。

【請求項 1 8】 中央処理装置を含む情報端末装置から当該情報端末装置で実行されるプログラムにおける所定の命令コードのアドレス情報を受け取り、受け取ったアドレス情報に基づいて前記情報端末装置の内部情報を取り込み、取り込んだ内部情報を用いて、そのアドレス情報に対応して規定されている処理を実行するものであることを特徴とする情報処理装置。

【請求項 1 9】 前記取り込んだアドレス情報に基づいて、取り込むべき前記情報のアドレスを求めることを特徴とする請求項 1 8 記載の情報処理装置。

【請求項 2 0】 前記所定の命令コードは割り込み命令を表すコードであることを特徴とする請求項 1 9 記載の情報処理装置。

【請求項 2 1】 特定の命令コードを含むプログラムを格納し、中央処理装置による前記特定の命令コードの実行によって、当該命令コードのアドレスを外部へ送信することを特徴とする情報端末装置。

【請求項 2 2】 前記特定の命令コードは、割り込み命令を表す命令コードであることを特徴とする請求項 2 1 記載の情報端末装置。

【請求項 2 3】 前記アドレスの外部送信に応答して外部から供給される外部要求を入力し、入力した外部要求に応答して所定の内部情報を外部へ提供することを特徴とする請求項 2 1 又は 2 2 記載の情報端末装置。

【請求項 2 4】 伝送路に接続可能なホスト装置を有する情報処理システムであって、前記ホスト装置は、アドレス情報に対応した機能を実行可能な実行ユニットを有し、プログラムを格納した情報端末装置から出力されるところの前記プログラムのメモリ空間におけるアドレス情報を受信し、受信したアドレス情報に対応した機能を提供するものであることを特徴とする情報処理システム。

【請求項 2 5】 前記実行ユニットは、複数のアドレス情報にそれぞれ対応した機能を提供可能とされていることを特徴とする請求項 2 4 記載の情報処理システム。

【請求項 2 6】 前記実行ユニットは、受信したアドレス情報に基づいて前記情報端末装置へのアクセス情報を形成することを特徴とする請求項 2 5 記載の情報処理システム。

【請求項 2 7】 前記プログラムには、割り込み命令を表すコードが設けられ、該割り込み命令を表すコードの実行により、前記アドレス情報を送信することを特徴とする請求項 2 4 乃至 2 6 の何れか 1 項記載の情報処理システム。

【請求項 2 8】 伝送路に接続可能なホスト装置を有する情報処理システムであって、前記ホスト装置は、互いに関連する第 1 のプログラムと第 2 のプログラムの内、第 1 のプログラムを端末装置へ提供し、該端末装置からの要求に応答して、前記第 2 のプログラムを実行することを特徴とする情報処理システム。

【請求項 2 9】 前記第 1 のプログラムは、前記情報処理システムから前記情報端末装置へ送信されることを特徴とする請求項 2 8 記載の情報処理システム。

【請求項 3 0】 前記第 1 のプログラムと前記第 2 のプログラムは、共通のソースプログラムより形成されるものであることを特徴とする請求項 2 8 又は 2 9 記載の情報処理システム。

【請求項 3 1】 前記要求は、前記第 1 のプログラムにおけるアドレス情報を含んでいることを特徴とする請求項 2 8 乃至 3 0 の何れか 1 項記載の情報処理システム。

【請求項 3 2】 情報端末装置に接続可能なホスト装置を有する情報処理システムであって、前記ホスト装置は、第 1 の処理をホスト装置へ要求するところのプログラムを情報端末装置へ提供し、情報端末装置からの要求に応じて前記第 1 の処理を実行することを特徴とする情報処理システム。

【請求項 3 3】 前記プログラムは、伝送路へ向けて提供されることを特徴とする請求項 3 2 記載の情報処理システム。

【請求項 3 4】 前記要求は伝送路から供給されることを特徴とする請求項 3 2 又は 3 3 記載の情報処理システム。

【請求項 3 5】 前記情報端末装置へ提供されるべき前記プログラムと、前記第 1 の処理に対応するプログラムとは、共通のソースプログラムより形成されたものであることを特徴とする請求項 3 4 記載の情報処理システム。

【請求項 3 6】 ネットワークを介してサーバーがクライアントにサービスを提供する情報処理方法であって、サーバーは、プログラムを格納した情報端末装置から出力されるところの前記プログラムのアドレス空間におけるアドレス情報をネットワークを介して受信し、受信したアドレス情報に対応するサービスの要求に応答するための処理を実行ユニットで実行し、実行結果に基づくサービスをネットワークを介してサーバーに送信することを特徴とする情報処理方法。

【請求項 3 7】 前記実行ユニットは、複数のアドレス情報にそれぞれ対応する処理の中から、受信したアドレス情報に対応する処理を選択して、その処理を実行することを特徴とする請求項 3 6 記載の情報処理方法。

【請求項 3 8】 前記実行ユニットによる処理の実行は、受信したアドレス情報に基づいて前記クライアントから読み込むべき情報をアクセスするためのアクセス情報の生成を含むことを特徴とする請求項 3 7 記載の情報処理方法。

【請求項 3 9】 前記プログラムには割り込み命令を表すコードが設けられ、該割り込み命令を表すコードの実行により、前記アドレス情報をネットワークに送信することを特徴とする請求項 3 6 乃至 3 8 の何れか 1 項記載の情報処理方法。

【請求項 4 0】 ネットワークを介してサーバーがクライアントにサービスを提供する情報処理方法であって、サーバーは、関連する第 1 のプログラムと第 2 のプログラムの内、第 1 のプログラムをネットワークを介してクライアントへ送信し、該クライアントからの要求に応答して、前記第 2 のプログラムを実行することを特徴とする情報処理方法。

【請求項 4 1】 前記第 1 のプログラムと前記第 2 のプログラムは、共通のソースプログラムより形成されるものであることを特徴とする請求項 4 0 記載の情報処理方法。

【請求項 4 2】 前記要求は、前記第 1 のプログラムのアドレス空間におけるアドレス情報を含んでいることを特徴とする請求項 4 0 又は 4 1 記載の情報処理方法。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は C++ 言語等の高級言語を用いて組込みシステムを開発するのに最適なシステム開発方法、そのためのプログラムを格納した記憶媒体、更には、ネットワークを介したサービスの提供を受けるクライアント若しくは情報端末装置における処理負担の軽減、更には、そのような負担軽減を考慮したサービス提供のための情報処理システム及び情報処理方法に関する。

【0 0 0 2】

【従来の技術】

計算機（本明細書においてコンピュータ装置などの電子計算機を単に計算機と称する）の開発言語である C++、Fortran 90、Java 等の高級言語が持つ高機能（例外処理など）は大きなリソース（スタック、メモリ等）を必要とする。このため、搭載メモリなどに制限の多い組込みシステム若しくは携帯情報端末装置等の開発言語には C++ のような高級言語に対するサブセット言語が用いられている。例えば EC++ という C++ のサブセット言語が提案されている。このようなサブセット言語を用いるときは、C++ が持つ例外処理機能等を含まないサブセットの範囲でソースプログラムを記述し、サブセット言語の処理系でコンパイルする。これにより、従来の C 言語と同程度のリソース効率で高級言語仕様を実現できる。

【0 0 0 3】

【発明が解決しようとする課題】

しかしながら、サブセット言語ではプログラムの標準化が困難である。すなわち、パーソナルコンピュータやワークステーション上で開発したプログラムを組み込み用途のマイクロコンピュータ等に移植することが困難になる。また、サブセットでは、プログラムデバッグ中だけに使用したい言語機能（例えば入出力、

例外処理等)が完備していないため、フルセットの開発言語環境と同様のデバッグを行なうことができず、デバッグ効率が悪くなる、という問題点のあることが本発明者によって明らかにされた。

【 0 0 0 4 】

そこで本発明者はターゲット装置(例えば組み込み用途のマイクロコンピュータ等)にとってオーバーヘッドになり得る高級言語機能も含めてフルセットの高級言語機能をシステム開発に利用可能にするために、ターゲット装置に対する負荷の軽減を行なうことを検討した。特に、ここで検討している負荷の軽減は、マルチプロセッサ向けコンパイラのように、処理能力が均質若しくは同等のプロセッサを対象として処理を分散させるという技術、とは基本的な着眼点が相違する、ということは本発明者によって予め把握されている。本発明者による検討内容は高級言語機能単位で負荷の軽減を図ろうとするものである。

【 0 0 0 5 】

更に本発明者が検討している負荷の軽減とは、開発環境の整備、即ちサブセットの開発言語ではプログラムデバッグ中だけでも使用したい高級言語機能(例えば入出力、例外処理等)が完備していないからデバッグ効率も悪くなるという問題点の解決、を出発点としており、この意味で、ターゲット装置の負荷の軽減に対して、軽減されるべき処理をデバッガに担わせることを検討した。特に、デバッガはコンパイルの過程で生成されるデバッグ情報を利用することにより軽減されるべきターゲット装置の処理を容易に実現できる、という点に着目している。

【 0 0 0 6 】

本発明者、そのターゲット装置とデバッガシステムとの関係を、ターゲットシステムの開発環境だけでなく、実システムにも拡張する検討を加えた。特にホスト(サーバー)と端末装置(クライアント、ターゲット装置)とを備えるような情報処理システムでは端末装置のリソースに大きな制限のある場合が想定され、特に、端末装置がPDA(Personal Digital Assistants)のような携帯情報端末装置や携帯電話機では既にその傾向が著しい。このような情報処理システムにおいて端末装置の負荷を軽減することを考慮したサービス提供のための情報処理システム若しくは情報処理方法によるビジネスモデルについても本発明者は検討

した。

【 0 0 0 7 】

本発明の目的は、ターゲット装置にとってオーバーヘッドになり得る高級言語機能も含めてフルセットの高級言語機能を利用可能にするシステム開発方法を提供することにある。

【 0 0 0 8 】

本発明の別の目的はC++言語等の高級言語を用いて組込みシステムを開発するのに最適なシステム開発方法を提供することにある

本発明の更に別の目的は、前記システム開発方法によるシステム開発を容易化することができるプログラムを格納した記憶媒体を提供することにある。

【 0 0 0 9 】

本発明のその他の目的は、ネットワーク等を介したサービスの提供を受ける情報端末装置の処理負担軽減を実現する情報処理装置、情報端末装置、並びに情報処理システムを提供することにある。

【 0 0 1 0 】

本発明のその他の目的は、ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化、及びクライアントとの間の通信の簡素化を実現することにある。

【 0 0 1 1 】

本発明の前記並びにその他の目的と新規な特徴は本明細書の記述及び添付図面から明らかになるであろう。

【 0 0 1 2 】

【課題を解決するための手段】

本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。

【 0 0 1 3 】

〔 1 〕 先ず基本的な考え方を一例として説明する。リソースなどの点で開発対象のターゲット装置がサポートできない若しくはターゲット装置にサポートさせたくない高級言語の高機能対応部分（例えばC++の例外処理）が軽減対象になる

。ソースプログラムの内の当該高機能対応部分については、その機能をデバッガシステム（若しくはホスト装置）で実現する為の第1コードに変換すると共に、デバッガシステム（若しくはホスト装置）にその第1コードを処理させるための要求等を発生させるターゲット用の第2コードに変換する。上記変換は例えばコンパイラ等を利用して行なわれる。更に、ターゲット装置のメモリ及びレジスタをアクセス可能にするモニタ機能として、デバッガシステム（若しくはホスト）用のモニタコマンド列（クライアントアクセスコマンド列）を生成する。モニタコマンド列は第1コードと同様に、第2コードの実行に応答して起動される性質のコードである。

【0014】

上記変換若しくはコンパイルされたオブジェクトプログラムをデバッガを用いてターゲット装置に実行させてソースプログラムの検証を行なうとき、ターゲット装置は高級言語の前記高機能対応部分では前記第2コードを実行する。第2コードは例えばTRAP命令のような割り込み命令のコードであり、高級言語の前記高機能対応部分を実行したとするときのオーバヘッドはない。第2コードがTRAP命令であればその命令コード固有のベクタ若しくは分岐先アドレスによって決まる割り込みハンドラを実行して、ターゲット装置はデバッガと通信を行ない、そのTRAP命令のアドレス（PCアドレス）をデバッガに与える。デバッガは、与えられたTRAP命令のアドレスに応ずるターゲット上のスタック領域の情報などを取り込み、前記与えられたTRAP命令のアドレスに対応する第1コードを、前記取り込んだスタック領域等の情報を用いて実行する。このようにして、ホスト装置が前記ターゲット装置のための前記高機能対応部分を実行若しくは模擬する。これにより、高級言語の機能単位に着目してターゲット装置の負荷を軽減することができる。デバッガによる前記スタック領域等に関する情報取り込みはモニタコマンド列を実行して行なう。モニタコマンド列によるアクセス機能は、前記割り込みハンドラの機能を介在させて実現してよい。

【0015】

前記高級言語の高機能対応部分が専らプログラムデバッグ中だけに使用したい例外処理などの言語機能である場合、デバッグ完了後、最終のオブジェクトプロ

グラム中の T R A P 命令のような第 2 コードを N O P に置き換えたりすればよい。この場合、ターゲット装置は、スタンドアロンシステムとしてホストから切り離して動作可能である。

【 0 0 1 6 】

最終プログラムに前記第 2 コードを残す必要がある場合、例えば実システムにおいてネットワーク上でクライアントの負荷を軽減することを想定するときは、最終プログラムに T R A P 命令のような第 2 コードを残し、ホスト装置には前記第 1 コードやモニタコマンド列等を組み込み、最終システム稼動後も、高級言語の高機能部分であるオーバーヘッド部分をネットワークのホスト装置に実行させることが可能になる。

【 0 0 1 7 】

〔 2 〕 《システムの開発方法》 先ず本発明をシステム開発方法の観点から具体的に説明する。

【 0 0 1 8 】

システム開発方法は、①一つの開発言語で記述され複数の記述を有するソースプログラム（ 4 ）における第 1 の記述を、第 1 の計算機（ 1 ）で実行可能な第 1 コードに変換し、②前記第 1 の記述を、該第 1 の記述によって定義される機能とは異なる機能を表すところのコードであって、第 2 の計算機（ 3 ）で実行可能な第 2 コード（ T R A P 命令コード）に変換すると共に、前記ソースプログラムにおける第 2 の記述を、前記第 2 の計算機で実行可能なその他のコード（第 3 コード）に変換し、③前記ソースプログラムのデバッグにおいて前記第 2 の計算機による前記第 2 コードの実行に応答して前記第 1 の計算機が前記第 1 コードを実行することを特徴とする。これにより、第 2 の計算機にとってオーバーヘッドになると考えられる高級言語の高機能部分（第 1 の記述）単位で第 1 の計算機にその処理を負わせることが可能になる。要するに、第 2 の計算機にとってオーバーヘッドになると考えられる高級言語の高機能部分（第 1 の記述）の処理を、第 2 計算機のリソースを用いることなく実現できる。

【 0 0 1 9 】

前記第 1 の計算機は、前記第 1 コードを実行する前に前記第 2 計算機から情報

を取り込む。例えば、第 1 の計算機は前記モニタコマンド列を実行し、第 2 のコードを実行したときの第 2 の計算機の内部状態、例えばスタック領域の状態等を取得する。第 1 の計算機は、前記スタック領域の情報等を参照し、第 1 コードを実行することにより、前記第 2 の計算機にとってオーバーヘッドになると考えられる高級言語の高機能部分（第 1 の記述）を模擬する。

【 0 0 2 0 】

前記第 2 コードは T R A P のような割り込み命令を表すコードである。第 2 の計算機は、当該割り込みコードの実行により割り込みハンドラによる処理に分岐する。割り込みハンドラは、例えば前記第 1 の計算機へ、前記第 2 コードのアドレス情報、例えば前記割り込み命令の命令アドレスを供給する処理を規定する。割り込みハンドラは、例えば第 2 計算機の O S （オペレーティングシステム）の一部に組み込まれ、或いはデバッグモニタプログラムとして第 2 のコードにリンクさせることができる。

【 0 0 2 1 】

前記第 2 コードのアドレス情報と前記第 1 コードとを対応させる対照データを第 1 の計算機で参照可能に生成しておく。これにより、第 1 計算機は、第 2 計算機による第 2 コードの実行に応答して実行すべき第 1 コードを複数の第 1 コードの中から簡単に選べる。

【 0 0 2 2 】

前記第 1 コードへの変換、第 2 コードへの変換、及びその他のコードへの変換はコンパイラを用いて行なえばよい。前記ソースプログラムのデバッグはデバッグを用いて行なえばよい。

【 0 0 2 3 】

開発されるべきシステムに着目したとき、ターゲット装置が単独の開発対象として位置付けられる場合、前記第 1 の計算機はデバッガシステムを構成する計算機であり、前記第 2 の計算機はターゲット装置を構成する計算機になる。このときの実システムにおいてソースプログラムの実行はターゲット装置単独で可能である。換言すれば、前記高級言語の高機能対応部分が専らプログラムデバッグ中だけに使用したい例外処理などの言語機能であて、実システムでは実行させる必

要がない場合、デバッグ完了後、最終のオブジェクトプログラム中の T R A P 命令のような第 2 コードを例えば N O P に置き換えて、ターゲット装置は、スタンダードアロンシステムとしてホスト装置から切り離して動作可能になる。

【 0 0 2 4 】

一方、実システムにおいて例えばネットワーク上でクライアントのようなターゲット装置の負荷を軽減することを想定するときは、最終プログラムに T R A P 命令のような第 2 コードを残し、ホスト装置には第 1 コード及びアクセスコマンド列を組み込み、最終システム稼動後も、高級言語の高機能部分であるオーバーヘッド部分をネットワークのホスト装置に実行させることが可能になる。このような場合、前記第 1 の計算機にはネットワークのホスト装置を構成する計算機を想定し、前記第 2 の計算機はターゲット装置である情報端末装置を構成する計算機を想定する。

【 0 0 2 5 】

〔 3 〕 《プログラムの記憶媒体》前記システム開発方法の実現に利用されるプログラムを格納した記憶媒体の観点より本発明を説明する。

【 0 0 2 6 】

記憶媒体 (3 1 , 3 2) は計算機によって読取り可能にプログラムを記録する。記録されたプログラムは、第 1 の記述と第 2 の記述を含むソースプログラムを入力する入力ステップ (S 1) と、入力した第 1 の記述を第 1 の計算機によって実行可能なコードに変換する第 1 変換ステップ (S 6) と、入力した前記第 1 の記述を該第 1 の記述により表される機能とは異なる機能であって、第 2 の計算機によって実行可能なコードへ変換する第 2 変換ステップ (S 4) と、入力した第 2 の記述を前記第 2 の計算機によって実行可能なコードに変換する第 3 変換ステップ (S 3) と、を実行させる。この記憶媒体を計算機に読取らせてコンパイラシステムやデバッガシステムを構成すると、前述のシステム開発方法を容易に実現することができる。

【 0 0 2 7 】

前記プログラムは、1 個の記憶媒体に格納しても、或いは複数個の記憶媒体に分散させても、或いは別のデータやプログラムと一緒に記憶媒体に記録してもよ

い。

【 0 0 2 8 】

前記プログラムを記憶媒体に取り込む手段は何ら限定されず、CD-ROMなどの様にプラスチック成形により、ハードディスクやフロッピーディスクなどのように磁気的変化によって、或いは、伝送線を介して記憶媒体に取り込まれてよい。

【 0 0 2 9 】

前記第2の計算機によって実行可能なコードが、割り込みを表す割り込みコードであるなら、前記プログラムは、前記割り込みコードにリンクされる割り込みハンドラを規定するステップを更に含んでよい。その割り込みハンドラを前記第2の計算機のオペレーティングシステムがサポートする場合にはリンクされる割り込みハンドラの内容を一々規定しなくてよい。

【 0 0 3 0 】

前記プログラムは、前記第2の計算機における前記割り込みコードのアドレスと前記第1の計算機が実行可能なコードとの対応を示すテーブルを形成するステップを更に含んでよい。

【 0 0 3 1 】

前記プログラムを読み取って実行する計算機はコンパイラシステムやデバッガシステムを構成する計算機である。このとき、前記第1の計算機はデバッガシステムを構成する計算機であり、前記第2の計算機はターゲット装置を構成する計算機である。開発システムのネットワーク利用を予め想定しているとき、前記第1の計算機はホスト装置を構成する計算機であり、前記第2の計算機は情報端末装置を構成する計算機である。

【 0 0 3 2 】

〔4〕《ホスト装置》前記システム開発方法等によって開発されるターゲットのための前記高機能対応部分を実行若しくは模擬するホスト装置のような情報処理装置の観点より本発明を説明する。

【 0 0 3 3 】

情報処理装置（40）は、中央処理装置（50）を含む情報端末装置（41）

から当該情報端末装置で実行されるプログラムにおける所定の命令コード例えば割り込み命令を表す命令コードのアドレス情報を受け取り、受け取ったアドレス情報に基づいて前記情報端末装置の内部情報を取り込み、取り込んだ内部情報を用いて、そのアドレス情報に対応して規定されている処理を実行する。この情報処理装置は、前記システム開発方法によって開発されるところの第2の計算機を有するターゲット装置を用いる実システムにおけるホスト装置に最適であり、ターゲット装置とされた情報端末装置におけるオーバーヘッド機能を代替することができる。

【 0 0 3 4 】

情報端末装置は前記内部情報を取り込むために、前記取り込んだアドレス情報に基づいて、取り込むべき前記内部情報のアドレスを求めれば、端末装置に負担をかけずに済む。

【 0 0 3 5 】

〔5〕《情報端末装置》前記システム開発方法等によって開発されるターゲットの観点より本発明を説明する。

【 0 0 3 6 】

前記ターゲットのような情報端末装置（41）は、割り込み命令を表す命令コードのような特定の命令コードを含むプログラムを格納し、中央処理装置（50）による前記特定の命令コードの実行によって、当該命令コードのアドレスを外部へ送信する。前記アドレスの外部送信に応答して外部から供給される外部要求を入力し、入力した外部要求に応答して所定の内部情報を外部へ提供する。割り込み命令のような命令を実行するだけで、情報端末装置がそのプログラムメモリ空間上のどの命令実行状態にあるかをホスト装置に連絡でき、ホスト装置は、そのアドレスに対応する処理を行なうことによって、情報端末装置にとってオーバーヘッドの大きな処理を代わって実行することができる。

【 0 0 3 7 】

〔6〕《情報処理システム》伝送路を介してクライアント若しくは情報端末装置にサービスを提供する情報処理システムの観点から本発明を説明する。

【0038】

サービスの要求を伝送路若しくはその集合体であるネットを介して受け、そのサービスを提供するサーバーを有するような情報処理システムを想定したとき、クライアントからサーバーのプログラムを呼び出すのではなく、サーバーはクライアントの命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供する。即ち、伝送路に接続可能なホスト装置（40）を有する情報処理システムにおいて、ホスト装置は、アドレス情報に対応した機能を実行可能な実行ユニット（53P, 53M）を有し、プログラムを格納した情報端末装置（41, 42, 43）から出力されるところの前記プログラムのメモリ空間におけるアドレス情報を受信し、受信したアドレス情報に対応した機能を情報端末装置に提供する。このように、ホスト装置（サーバー）は情報端末装置（クライアント）の命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供するというプロトコルを採用するから、ホスト装置と情報端末装置との間の通信処理が簡素であり、また、情報端末装置において前記アドレス情報を出力する処理を規定するプログラムとホスト装置においてアドレス情報に応じた処理を行なうプログラムを前述のシステム開発方法を適用して形成することができる。したがって、ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化も実現できる。

【0039】

前記実行ユニットは、複数のアドレス情報にそれぞれ対応した機能を提供可能とされる。前記実行ユニットは、受信したアドレス情報に基づいて前記情報端末装置をアクセスするためのアクセス情報を形成したりしてよい。

【0040】

前記プログラムには、割り込み命令を表すコードが設けられ、該割り込み命令を表すコードの実行により、前記アドレス情報を送信してよい。

【0041】

〔7〕《情報処理システム》伝送路を介してプログラムを提供するような情報処理システムを想定する。即ち、インターネット等を介するプログラムのダウンロード、或いは、クライアント内で作られたプログラムにおいて使われるソフト部

品をサーバーが提供するようなシステムを想定する。このとき、クライアントのリソースの観点より、全てのプログラムを自分で実行することが困難若しくは不経済な場合がある。これに着目して、クライアントが欲するプログラムを、少なくとも2つに分け、サーバーは、一方をクライアントに提供し、クライアントからの要求で他方のプログラムを実行する。例えば、通信ネットワークに接続可能なホスト装置（40）を有する情報処理システムであって、前記ホスト装置は、一体として使われるべき第1のプログラム（PGM1）と第2のプログラム（PGM2）の内、第1のプログラムを情報端末装置（41，42，43）へ提供し、該端末装置からの要求に応答して、前記第2のプログラムを実行する。

【0042】

前記第1のプログラムは、前記情報処理システムから前記情報端末装置へ送信されてよい。前記第1のプログラムと前記第2のプログラムは、共通のソースプログラムより形成されてよい。この場合には前述のシステム開発方法を適用可能である。前記要求は、前記第1のプログラムにおけるアドレス情報を含んよい。

【0043】

〔8〕《情報処理システム》クライアントによる処理のオーバーヘッドを解決するという観点に立ち返れば、情報端末装置に接続可能なホスト装置（40）を有する情報処理システムにおいて、前記ホスト装置は、第1の処理をホスト装置へ要求するところのプログラムを情報端末装置（41，42，43）へ提供し、情報端末装置からの要求に応じて前記第1の処理を実行する。前記プログラムは、伝送路（44）へ向けて提供され、前記要求は伝送路から供給される。システムのコスト低減のためには、前述の通り、情報端末装置へ提供されるべき前記プログラムと、前記第1の処理に対応するプログラムとを、共通のソースプログラムより形成するとよい。

【0044】

〔9〕《情報処理方法》ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供する情報処理方法の観点に立って本発明を説明する。

【0045】

前述の情報処理システムの観点と同様に、サービスの要求をネットを介して受

けることを想定したとき、クライアントからサーバーのプログラムを呼び出すのではなく、サーバーはクライアントの命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供する。即ち、ネットワークを介してサーバーがクライアントにサービスを提供する情報処理方法であって、サーバーは、プログラムを格納した情報端末装置から出力されるところの前記プログラムのアドレス空間におけるアドレス情報をネットワークを介して受信し、受信したアドレス情報に対応するサービスの要求に応答するための処理を実行ユニットで実行し、実行結果に基づくサービスをネットワークを介してサーバーに送信する。このように、サーバーはクライアントの命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供するというプロトコルを採用するから、ホストとクライアントとの間の通信処理が簡単であり、また、ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化も実現できる。

【 0 0 4 6 】

また、ネットを介してプログラムを提供するような情報処理方法、例えば前述のように、インターネット等を介するプログラムのダウンロード、或いは、クライアント内で作られたプログラムにおいて使われるソフト部品をサーバーが提供するような方法を想定する。このとき、クライアントのリソースの観点より全てのプログラムを自分で実行することが困難若しくは不経済な場合がある。そこで、ネットワークを介してサーバーがクライアントにサービスを提供する情報処理方法では、サーバーは、一体として使われるべき第1のプログラムと第2のプログラムの内、第1のプログラムをネットワークを介してクライアントへ送信し、該クライアントからの要求に応答して、前記第2のプログラムを実行する

【 0 0 4 7 】

【発明の実施の形態】

《システム開発方法》

本発明に係るシステム開発方法の具体例を説明する。ここでは、組み込み制御にマイクロコンピュータを用いるターゲットマシンをC++高級言語を開発言語として開発する場合について説明する。

【 0 0 4 8 】

図 2 にはターゲットマシンの開発に用いるデータ処理システムが例示される。ホストマシン 1 はパーソナルコンピュータ若しくはエンジニアリングワークステーション等の計算機とされ、伝送路若しくはネットワーク 2 を介してホストマシン 1 にターゲットマシン 3 が接続される。ホストマシン 1 は、そこに組み込まれているプログラム及びデータによって機能が実現され、ターゲットマシンの開発段階においてはコンパイラシステム、デバッガシステムなどの開発環境を構成している。特に説明するまでもないが、ホストマシン 1 は、プロセッサ、RAM、ROM 及びインタフェースコントローラ等が実装され、インタフェースコントローラにはディスプレイ、キーボード、ハードディスクなどの補助記憶装置が接続され、プロセッサの動作プログラムに従ってデータ処理可能にされる。

【 0 0 4 9 】

ターゲットマシン 3 の開発に当り、ターゲットマシンに実装されるマイクロコンピュータが実行するプログラムは、先ずフルセットの C++ 高級言語によってソースプログラムとして記述される。

【 0 0 5 0 】

図 1 にはシステム開発方法で利用される各種ソフトウェアの関係が示される。コンパイラ 5 はソースプログラム 4 をデバッグスクリプト 6 とロードモジュール 7 に変換する。ロードモジュール 7 はターゲットマシン 3 のアプリケーションプログラム 8 及びデバッグモニタプログラム 9 としてターゲットマシン 3 にダウンロードされる。デバッグスクリプト 6 は、ホストマシン 1 上でデバッガ 10 及び通信機能プログラム 11 として実行される。

【 0 0 5 1 】

前記コンパイラ 5 は、フルセットの C++ 言語で記述されたソースプログラム 4 をターゲットマシン 3 のマイクロコンピュータ用のオブジェクトプログラムにコンパイルする機能を有する。このとき、C++ 高級言語機能のうち、リソース（メモリ、入出力機能等）の限られたマイクロコンピュータにおいてオーバーヘッドなく実現できる範囲、例えば J I S の C 言語仕様（J I S 規格「プログラム言語 C」）における「フリースタANDING仕様」の範囲の記述（第 2 の記述）

については、前記マイクロコンピュータ上のオブジェクトコード（第 3 コード若しくはその他のコード）に変換する。ソースプログラムにおけるそれ以外の処理を規定する記述（第 1 の記述）、即ち例外処理などの高級言語の高機能記述に対しては、ホストマシン 1 で実行可能な命令若しくはコマンド列（第 1 コード）に変換し、また、ターゲットマシン 3 側で実行可能なオブジェクトコードとして、例えば所定の T R A P 命令のコード（第 2 コード）に変換する。第 1 コードは前記第 1 の記述によって定義される機能をホストマシン 1 上で実現するコードである。これに対して第 2 コードは、第 1 の記述によって定義される機能とは異なる機能をターゲットマシン 3 上で実現するコードであればよい。要するに、第 2 コードにはターゲットマシン 3 に負荷をかけずに実行可能なコードを選べばよい。このように、コンパイラは、高級言語の機能単位に着目してターゲットマシン 3 に負荷をかける言語機能単位をホストマシン 1 で実行可能にするものであり、高級言語の機能単位に着目してターゲットマシン 3 の負荷を軽減させる。

【 0 0 5 2 】

ターゲットマシン 3 における前記第 2 コードの実行によってホストマシン 1 に第 1 コードを実行させることを想定する。そのためには、デバッガ 1 0 とターゲットマシン 3 との間での連絡が必要になり、それには以下の最低限の機能があればよい。最低限の機能は、①ターゲットマシン 3 からデバッガ 1 0 への処理の要求、②デバッガ 1 0 によるターゲットマシン 3 のメモリへの読み込み及び書き込み、③デバッガ 1 0 によるターゲットマシン 3 のレジスタへの読み込み及び書き込み、である。これは、ターゲットマシン 3 のマイクロコンピュータのためのデバッグモニタプログラムによって実現できる機能である。

【 0 0 5 3 】

ここでは、ターゲットマシン側ではそれらの機能①②③を、ターゲットマシン 3 上のマイクロコンピュータ（ターゲットマイコン）のアプリケーションプログラム内で T R A P 命令を発行することによって起動される割り込みハンドラの処理を介して支援する。このとき、デバッガ側の機能として必要なのは、各 T R A P 命令の命令アドレスに対応する第 1 コードで規定される高級言語機能の実現、次に、T R A P 命令の置かれた命令アドレスの情報取得、そして、高級言語機能

をデバッガ側でシミュレートするために必要な、変数とそのターゲットマシン 3 のメモリ上のアドレスとの対応情報（その時点におけるスタックの状態を示す情報など）の取得である。この点において割り込みハンドラは、ターゲットマシン 3 上における T R A P 命令コードの命令アドレスを出力する機能を有する。さらに、コンパイラ 5 は、デバッガ 1 0 側において上記②、③の連絡機能を用いてデバッガ 1 0 側でターゲットマシン 3 のメモリ及びレジスタの内容を操作可能とするデバッグコマンド列を生成する。デバッグコマンド列及び第 1 コードは対応する T R A P 命令の命令アドレスをキーとしてリンクできる対照表若しくは対照テーブルを構成している。

【 0 0 5 4 】

尚、T R A P 命令コードは割り込み命令を表すコードであり、ターゲットマシン 3 におけるマイクロコンピュータのアーキテクチャ上、規定のベクタ等で指し示される割り込みハンドラによる処理に分岐させる命令である。割り込みハンドラはターゲットマシン 3 の O S （オペレーティングシステム）でサポートしてもよい。或いはコンパイラ 5 がその T R A P 命令コードに対応する規定の割り込みハンドラをその T R A P 命令コードにリンクさせてもよい。図 1 において割り込みハンドラは、デバッグモニタプログラム 9 の一部として図示されているが、それに限定されない。

【 0 0 5 5 】

図 3 には前記コンパイラ 5 による第 1 の記述に関するコンパイル機能の詳細が例示される。ソースプログラムに含まれる高機能記述部分を抽出すると、その第 1 記述は、その記述で示される機能をホストシステムで模擬する為に必要なデバッグスクリプト 1 0, 1 1 に変換され、アプリケーションプログラム 8 としてのターゲットプログラムのための T R A P 命令コード（第 2 コード）に変換される。デバッグスクリプト 1 0, 1 1 は前記第 1 コード及びデバッグコマンド列を含み、対応する T R A P 命令コードの命令アドレスとの対照表を構成する。

【 0 0 5 6 】

コンパイラ 5 によりコンパイル処理されたプログラムをデバッグするとき、ホストマシン 1 はデバッガ 1 0 で動作され、ターゲットマシン 3 はアプリケーション

プログラム 8 で動作される。デバッグでは先ず、ターゲットマシン 3 のプログラムをデバッガ 1 0 を用いて起動する。

【 0 0 5 7 】

ターゲットマシン 3 が高級言語機能に相当する T R A P 命令を実行し、これによって、T R A P 命令発行時の命令アドレスがデバッガ 1 0 に与えられると、デバッガ 1 0 は、前記対照表により T R A P 命令発行時の P C (プログラムカウンタレジスタ、即ち T R A P 命令自身のアドレス) から、実行すべきデバッグコマンド列と第 1 コードをサーチして実行する。これにより、デバッガ 1 0 は T R A P 命令の命令アドレスを用いてその時のターゲットマシン 3 におけるスタック領域の情報などを取得し、第 1 コードで示される高級言語機能をデバッガ側で模擬することができる。このように、C++ の例外処理機能をターゲットマシン 3 に代わってデバッガ 1 0 がシミュレートすることにより、システムリソースの限られたターゲットマシン 3 を C++ のフルセット仕様の開発環境を用いて開発することが可能にある。

【 0 0 5 8 】

デバッグが終了し、その結果、ターゲットマシン 3 上では例外処理機能等の高機能が不用になれば、デバッガモニタプログラム、換言すれば T R A P 命令の割り込みハンドラを、単に T R A P 命令を無視するソフトウェアと切換えて最終ソフトウェアを構築すればよい。或いは、T R A P 命令を埋め込んだ命令アドレスが分かっているから、T R A P 命令を N O P (ノン・オペレーション) 命令に置き換えて、最終的なアプリケーションプログラムとしてもよい。このことから理解されるように、デバッガ 1 0 は、そのような T R A P 命令が発生したかどうかを表示できる機能を持つことが望ましい。

【 0 0 5 9 】

ターゲットマシン 3 を伝送路若しくはネットワーク上で実現する場合には、ターゲットマシン 3 の最終プログラムには T R A P 命令及びデバッグモニタプログラム (割り込みハンドラ) を含め、最終的に開発されたシステムの稼動後も、高級言語機能のオーバーヘッド部分を規定する第 1 コード及びデバッグコマンド列をホストマシンに残し、ターゲットマシン 3 が前記所定の T R A P 命令を実行し

たとき、デバッグ時と同様にその高級言語機能をホストマシンでシミュレートしてやればよい。

【 0 0 6 0 】

このように、オーバヘッドの大きな高級言語の機能を、ターゲットマシン上では T R A P 命令 1 命令で実現できるから、ターゲットマシンのオーバヘッドを最小にでき、リソースの少ないマイクロコンピュータ若しくはターゲットマシン上で高級言語によって開発されたプログラムを効率よく実行できる。

【 0 0 6 1 】

ホストマシンにターゲットマシンのオーバヘッド機能部分を負担させることにより、ターゲットマシンの価格を下げるができる。電話局と携帯電話のように、ホストマシン 1 個に多数のターゲットマシンが対応するシステムにおいて特に顕著である。ターゲットマシンのリソースが少なくても、高級言語をフルスペックで実装することができる。組込み用途のシングルチップマイクロコンピュータ、そのコンパイラやデバッガ、通信ネットワークで特に効果がある。

【 0 0 6 2 】

《オブジェクト解放処理の模擬》

前記 C ++ で記述されたターゲットマシンの為の例外処理をホストマシンで模擬するための具体例について説明する。

【 0 0 6 3 】

図 4 には C ++ の言語機能を模式的に示してある。図 4 では、特に制限されないが、C ++ 言語の全ての機能を、オーバーヘッドの大きな例外処理機能 (F 1) 、テンプレート・クラス機能 (F 2) 、C 言語機能 (F 3) に分類してある。ここでは、例外処理機能 (F 1) を T R A P 命令とデバッガスクリプトへの変換対象とする。その他の機能は、ターゲットマシンのオブジェクトコードへの変換対象とする。

【 0 0 6 4 】

前記例外処理機能を説明する。C ++ 言語における例外処理は、C a t c h 構文によって例外処理の有効となる範囲と例外処理時の操作を宣言し、t h r o w 文の実行により例外条件を発生させる。C a t c h 構文と t h r o w 文は必ずし

も同じ関数内に無くてもよい。C a t c h 構文及び t h r o w 文を用いた記述の簡略化された例は図 3 のソースプログラムの記述として示されている。

【 0 0 6 5 】

C++言語における例外処理の問題点は、例外処理が登録される関数と例外が発生する関数に対してオーバーヘッドとなる余分な処理が発生するだけでなく、その間で、（動的に）呼び出され得る全ての関数において例外発生の可能性に対して備えなければならないということである。

【 0 0 6 6 】

この点を更に明確化するために、C++言語における例外発生時の処理を説明する。例外が発生すると、①C a t c h 構文の例外処理の宣言のプログラム点に制御を移行し、②C a t c h 構文のある関数から t h r o w 文のある関数までに関数呼び出しによって割り付けられたスタック領域を解放し、③C a t c h 構文で宣言された処理を実行する。

【 0 0 6 7 】

このとき、C++言語はオブジェクト指向言語であるため、関数呼び出しによって割付けられたスタック領域を開放するとき、当該関数で作成したオブジェクトの解放処理を実施する必要がある。

【 0 0 6 8 】

C++言語におけるオブジェクトの解放処理を実現する通常行なわれている一方式を図 5 に基づいて説明する。図 5 の（A）の関数 f、g に対して割当てられたスタックフレームの一例が（B）に示されている。図 5 のスタックフレームにおける下位アドレスは紙面の下方である。親の関数 f のスタックフレームの上位側アドレスに子の関数 g のスタックフレームが形成されている。

【 0 0 6 9 】

〔I〕各関数のスタックフレームの最下位アドレスには 2 ワードの領域を確保し、1 ワード目には親の関数のフレームの先頭アドレス（フレームポインタ）、2 ワード目には当該関数のオブジェクトを開放するための関数のアドレス（解放処理関数のアドレス）を保持する。例えば図 5 の（B）に示される関数 g のスタックフレームでは、1 ワード目には g のフレームポインタ P 1 g が配置され、2 ワ

ード目には g の解放処理関数のアドレス $P_2 g$ が配置される。

【0070】

〔I I〕例外発生時に当該関数のオブジェクト解放処理実施後、スタックフレームが例外処理を登録した関数に達するまで、〔I I I〕のスタック解放処理を実施する。

【0071】

〔I I I〕子関数 g は解放処理関数のアドレスを呼び出すことにより自フレーム内のオブジェクト（オブジェクト0、オブジェクト1、…）の解放処理を実施し、更にフレームポインタ $P_1 g$ により親関数 f のスタック領域最下位アドレスを求め、 g のスタックフレームを全て開放する。

【0072】

しかし、例外処理をターゲットマシン自身が実行しないのであれば、図6に例示されるように、各スタックフレームの上位2ワードのフレームポインタ及び解放処理関数のアドレスのエリア（例えば $P_1 g$ 、 $P_2 g$ ）を省くことができる。これによって、貴重なオンチップRAMの領域を節約することができる。即ち、ターゲットマシン若しくはその実装マイクロコンピュータのリソースを節約することができる。換言すれば、マシンリソースに制限がある場合には、前記フレームポインタ及び解放処理関数のアドレスのエリアを省くために、前述のシステム開発方法で説明したようにして、例外処理をターゲットマシン自身で実行させなければよい。

【0073】

但し、それら2ワードは例外処理を登録する関数や例外処理を発生する関数において必要なだけでなく、これら登録と例外の間に（動的に）呼び出され得る全ての関数に必要である。したがって、C++言語ではC言語に対して全ての関数においてスタックフレーム内に2ワードの余分な領域が必要になる。異なる実現方法を採用したとしても、C++の例外処理という言語機能が全ての関数に対して影響を与え、C言語よりもリソースを必要とする機械語を要求することになり変わらない。

【 0 0 7 4 】

ターゲットマシン 3 上の R A M にフレームポインタ及び解放処理関数のアドレスのエリア（例えば P 1 g, P 2 g）を持たないで、例外処理時の前記〔 I 〕～〔 I I I 〕の操作を、実行するためには、デバッガは以下の処理を実行する。

【 0 0 7 5 】

〔 I V 〕 ターゲットマシンにおける T R A P 命令の命令アドレス（例外が発生したときのプログラムカウンタ）から、T R A P 命令を含む関数名と、関数内のプログラムカウンタの値を求める。

【 0 0 7 6 】

〔 V 〕 当該関数のデバッグ情報から、関数の出口（オブジェクト解放処理の直前）でのスタックポインタと現スタックポインタの値の差を求め（これはコンパイラが管理する情報であり、デバッグ情報に含めることができる）、スタックポインタを関数の出口における値に変更する。

【 0 0 7 7 】

〔 V I 〕 当該関数のデバッグ情報から、当該関数のオブジェクト解放処理関数のアドレスを求め、その関数を呼び出す。呼び出すためには解放処理関数の出口にブレークポイントを設け、プログラムカウンタの値を解放処理関数のエントリに設定して再実行すればよい。

【 0 0 7 8 】

〔 V I I 〕 当該関数にデバッグ情報から関数の出口でのスタックフレームのサイズを求め、親関数が当該関数を呼び出したところまでスタックを解放する。

【 0 0 7 9 】

〔 V I I I 〕 上記〔 I V 〕～〔 V I I 〕を、例外処理を登録した関数のスタックポインタの位置に達するまで繰り返す。

【 0 0 8 0 】

〔 I X 〕 尚、解放処理の中で例外が発生する可能性があるため、上記処理のネストをデバッガが管理する必要がある。

【 0 0 8 1 】

上記〔 I V 〕～〔 I X 〕の処理は図 1 で説明した前記第 1 コードに含まれてい

る。これら処理は、全てデバッガが持つデバッグ情報に基づいて実施するため、ターゲットマシンにおいて関数のスタックフレーム上にこの処理に必要な情報を置く必要はない。要するに、デバッガ10がターゲットマシン3の例外処理を代替する場合でも、スタックフレームに前記フレームポインタ及び解放処理関数のアドレスを格納する2ワードのエリアを設けることは一切不要になる。実際、〔V〕、〔V I I〕で用いるスタックフレームサイズや関数内でのプログラムカウンタの値に対するスタックポインタのオフセット値が図5の(B)におけるフレームポインタの役目を果たし、また、〔V I〕で用いる、当該関数のオブジェクト解放処理のアドレス(デバッグ情報内)が図5の(B)におけるスタックフレーム内のオブジェクト解放処理のアドレスの役目を果たすため、これら2ワードをターゲット装置のRAM上のスタックフレーム内に保持する必要は全くない。

【0082】

《システム開発方法》

次に、伝送路若しくはネットワークを介してサーバーがクライアントにサービスを提供する情報処理システムの観点に立ったシステム開発方法を説明する。

【0083】

図7にはクライアントのシステムを開発するときのシステム開発方法が例示される。クライアントのソースプログラムはフルセットのC++言語で記述される。システム開発装置はクライアントに実装されるマイクロコンピュータをサポートするコンパイラ及びデバッガを組み込んだパーソナルコンピュータ若しくはエンジニアリングワークステーション等の計算機が用いられる。

【0084】

システム開発装置はソースプログラムを入力する(S1)。入力されたソースプログラムの記述に対してサーバーで実行する記述であるか否かを判定する(S2)。例えば前述のJISのC言語仕様(JIS規格「プログラム言語C」)における「フリースタANDING仕様」の範囲の記述についてはクライアントが実行する記述(第2の記述)と判定し、ソースプログラムにおけるそれ以外の処理を規定する記述、即ち例外処理などの高級言語の高機能記述に対しては、サーバーが実行する記述(第1の記述)と判定する。或いは、高級言語の高機能記述に

加えて、前記「フリースタANDING仕様」の範囲の記述であっても処理時間が長いと予め考えられる処理、利用頻度が少ないと予め想定される処理例えば故障診断或いは自己救済機能に関する記述についても第1の記述と判定する。

【0085】

第2の記述に対しては、クライアント用のオブジェクトコード（第3コード）に変換する（S3）。前者の第1の記述に対しては、先ずその記述をクライアント用の前記TRAP命令コード（第2コード）に変換し（S4）、クライアントのメモリ空間における当該TRAP命令の命令アドレスを保持し（S5）、更に第1の記述が示す機能をサーバー用のオブジェクトコード（第1コード）変換し、クライアント内のメモリやレジスタをリード・ライトする為のデバッグコマンド列に相当するクライアントアクセス用コマンド列を生成する（S6）。最後に、S3及びS4で生成されたクライアント用オブジェクトプログラム20、S5乃至S7で生成されたサーバー用のオブジェクトプログラム及びクライアントアクセス用コマンド列21が出力される（S8）。前記サーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列21は、クライアント用オブジェクトプログラム20に含まれるTRAP命令コードの命令アドレスとリンクされた対照テーブルを構成する。

【0086】

このように、システム開発装置は、前述の図1の説明と同様に、クライアントに負荷をかけると考えられる処理をサーバーで実行可能にするものであり、クライアントのマシンリソースを考慮してその負荷を軽減させる。

【0087】

特にこの例では、クライアントの負荷を最小限とするために、TRAP命令による割り込みハンドラはクライアントの専用OSがサポートし、クライアントによるTRAP命令の実行により、当該TRAP命令のアドレスを出力し、その後、サーバーからのリード・ライト要求に答える通信制御を支援するようになっている。

【0088】

図8には図7の方法で生成されるプログラムが例示される。図8においてソー

スプログラム 2 2 は例えば P 1、P 2、P 3 の記述を含んでいる。図 7 の方法で出力されるクライアント用のオブジェクトプログラム 2 0 は P O 1 (A 1)、T R A P (A 2)、P O 3 (A 3) のオブジェクトコードを含む。A 1、A 2、A 3 はコードのマッピングアドレスを意味する。すなわち、クライアント内のマイクロコンピュータが管理するメモリ空間において、オブジェクトコードが配置されるところのアドレスを表している。命令コード P O 1、P O 3 はソース記述 P 1、P 3 が変換されたものである。図 7 の方法で出力されるサーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列 2 1 は、ソース記述 P 2 をサーバーのオブジェクトコードに対応させて変換したプログラム P O 2 とクライアントアクセス用コマンド列を T R A P 命令のアドレス A 2 でリンクさせたテーブル構造を有する。

【 0 0 8 9 】

図 9 には前記クライアントのシステムを開発するときのシステム開発装置 3 0 が示される。クライアントのソースプログラムはフルセットの C++ 言語で記述される。システム開発装置は前記クライアントに実装されるマイクロコンピュータをサポートするコンパイラ及びデバッガを組み込んだパーソナルコンピュータ若しくはエンジニアリングワークステーション等の計算機 3 0 が用いられる。コンパイラ及びデバッガ等の開発支援プログラムは単数又は複数毎のフロッピーディスク (F D) 3 1 や C D - R O M ディスク 3 2 等の記録媒体に原始的に記録されて提供され、それら記録媒体に記録されたプログラムは計算機内部のハードディスク装置 3 3 などの記録媒体にインストールされて保持される。或いは、そのようなプログラムは、公衆回線網 3 4 などを通じてダウンロードされ、前記ハードディスク装置 3 3 に記録されている。前記開発支援プログラムは、図 7 に示したステップを図 9 の計算機 3 0 に実行させる。

【 0 0 9 0 】

尚、図 1 乃至図 6 で説明したシステム開発方法に利用する前記コンパイラ 5 等のシステム開発プログラムも C D - R O M 3 2 などの記録媒体に記録されてシステム開発装置に提供される。この時のシステム開発装置は図 2 ではホストマシン 1 であり、ホストマシン 1 はその記録媒体からシステム開発プログラムを読み込

んで実行する。

【 0 0 9 1 】

《情報処理システム》

次に、伝送路若しくはネットワークを介してサーバーがクライアントにサービスを提供する情報処理システムを説明する。

【 0 0 9 2 】

図 1 0 には情報処理システムの一例が示される。情報処理システムは、ネットワーク 4 4 のサーバー 4 0 と複数のクライアント 4 1 ~ 4 3 が接続されて成る。ネットワーク 4 4 は、公衆回線網、携帯電話網、LAN（ローカルエリアネットワーク）、インタネット、或いはそれらの複数種類混在であってもよい。サーバー 4 0 はワークステーションやパーソナルコンピュータなどの計算機を有し、代表して記憶ユニット 4 0 S が例示される。クライアント 4 1 ~ 4 3 は携帯電話機、PDA などの携帯情報端末装置等であって、CPU のような計算機を有し、代表してメモリ 4 1 M ~ 4 3 M が示される。記憶ユニット 4 0 S、メモリ 4 1 M ~ 4 3 M はデータ及びプログラム格納エリアとして利用される。

【 0 0 9 3 】

クライアント 4 1 ~ 4 3 の動作プログラムは図 7 のシステム開発方法で得られたクライアント用オブジェクトプログラム 2 0 とされる。このクライアント用オブジェクトプログラム 2 0 は図 9 のシステム開発装置 3 0 から夫々のメモリ 4 1 M ~ 4 3 M に格納される。格納形態はクライアント 4 1 ~ 4 3 の製造段階で画一的に行ない、或いはネットワーク 4 4 にシステム開発装置 3 0 が接続されている場合にはクライアントが個別的にダウンロードで行なうことができる。格納先は個々のクライアント 4 1 ~ 4 3 のメモリ 4 1 M ~ 4 3 M である。前述の如く、クライアントにとってオーバーヘッドの大きな処理は TRAP 命令に置き換えてサーバー 4 0 に処理させるので、そのためのサーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列を含む対照テーブル 2 1 は前記システム開発装置 3 0 からサーバー 4 0 の記憶ユニット 4 0 S に格納される。この格納形態もクライアントと同様に製造段階或いはネットワーク経由の何れでもよい。

【 0 0 9 4 】

図 1 1 にはクライアント 4 1 の具体例が示される。クライアント 4 1 は、CPU（中央処理装置）5 0、入出力ユニット 5 1、及び通信ユニット 5 2 を有する。CPU 5 0 はフェッチした命令を解読して制御信号を生成する命令制御部と、その制御信号で制御される演算部を有し、フェッチした命令を実行する。図 1 1 では演算部のレジスタ 4 1 R 及びメモリ 4 1 M が代表的に例示される。メモリ 4 1 M は CPU の動作プログラム等を保有する例えば電氣的に書き換え可能な不揮発性メモリ、及び CPU のワーク領域とされる RAM を備える。入出力ユニット 5 1 はディスプレイ、キースイッチ、スピーカ、マイクロフォンなどを有し、マンマシンインタフェース機能を実現する。通信ユニットはクライアント 4 1 を外部とインタフェースする手段であり、例えば、携帯電話機であれば、高周波部、変復調処理部、A/D・D/A 変換部などを有し、携帯電話網を介して通信可能にされる。或いは、LAN アダプタ、MODEM アダプタ、シリアルインタフェース、パラレルインタフェースなどであってもよい。その他のクライアント 4 2、4 3 も同様に構成される。

【 0 0 9 5 】

図 1 2 には前記サーバー 4 0 の具体例が示される。サーバー 4 0 は、ワークステーションなどの計算機 5 3、通信ユニット 5 4、及び前記記憶ユニット 4 0 S を有する。計算機 5 3 は例えば実行ユニットとしてマイクロプロセッサ 5 3 P 及びマイクロプロセッサ 5 3 P のワーク領域として利用される RAM 5 3 M を有する。計算機 5 3 はその他に図示を省略するマンマシンインタフェース回路などを備えている。記憶ユニット 4 0 S はサーバー用プログラム領域と共にクライアント用プログラム領域を有する。クライアント用プログラムは、前記サーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列を含む対照テーブル 2 1 を格納する領域である。サーバーとしての固有の動作の為の動作プログラムはサーバー用プログラム領域に格納される。通信ユニット 5 4 はネットワーク 4 4 に接続可能なルータ、モデム、LAN 等の通信機能を有する。

【 0 0 9 6 】

図 1 3 にはクライアント 4 1 による前記クライアント用オブジェクトプログラ

ム 2 0 の実行制御状態が示される。

【 0 0 9 7 】

前記クライアント 4 1 は、前記 T R A P 命令コードを含むクライアント用オブジェクトプログラム 2 0 をメモリ 4 1 M に保有し、C P U 5 0 がそのクライアント用オブジェクトプログラム中の T R A P 命令コードを実行すると、対応する割り込みハンドラが起動され、当該命令コードのアドレス（C P U 5 0 が管理するメモリ空間におけるアドレス）を通信ユニット 5 2 を介して外部へ送信する（S 1 0）。前記アドレスの外部送信に応答してサーバー 4 0 から、その時のスタック領域の状態などの出力要求があると、これを入力し（S 1 1）、入力した要求に応答して、スタック領域の状態等の内部情報をアクセスしてサーバー 4 0 に提供する（S 1 2）。T R A P 命令のような命令を実行するだけで、クライアント 4 1 がそのプログラムメモリ空間上のどの命令実行状態にあるかをサーバー 4 0 に連絡でき、サーバー 4 0 は、そのアドレスに対応する処理を行なうことによって、クライアント 4 1 にとってオーバヘッドの大きな処理を代わって実行することができる。

【 0 0 9 8 】

図 1 4 にはサーバー 4 0 による前記対照テーブル 2 1 を用いたプログラム及びコマンド実行制御状態が示される。

【 0 0 9 9 】

サーバー 4 0 の計算機 5 3 は、C P U 5 0 を含むクライアント 4 1 から当該クライアントで実行されるプログラム中の所定の命令コード例えば T R A P 命令コードのアドレス情報（アドレス P C）を受け取り、これによって、前記対照テーブル 2 1 のプログラム実行要求のあることを判定する（S 2 0）。計算機 5 3 は、受け取ったアドレス情報をキーとして対照テーブル 2 1 をサーチし、そのアドレス情報にリンクされているサーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列を記憶ユニット 4 0 S から R A M 5 3 M にロードする（S 2 1）。計算機 5 3 はクライアントアクセス用コマンド列を実行し、そのアドレス情報に基づいてその時の前記クライアント 4 1 のスタック領域の状態等の内部情報を取り込み（S 2 2）、取り込んだ内部情報を用いて、そのアドレス情報

にリンクされている対照テーブル 2 1 のサーバー用オブジェクトプログラムを実行する (S 2 3)。このサーバー 4 0 は、クライアント 4 1 におけるオーバーヘッド機能を代替することができる。サーバー 4 0 は、前記内部情報を取り込むために、前記取り込んだアドレス情報に基づいて、取り込むべき前記内部情報のアドレスを求める処理を、前記対照テーブル 2 1 のクライアントアクセス用コマンド列を実行して生成するので、この点においてもクライアント 4 1 に負担をかけない。

【 0 1 0 0 】

《情報処理方法》

図 1 0 の情報処理システムにおいてサービスの要求をネットワーク 4 4 を介して受けることを想定したとき、クライアント 4 1, 4 2, 4 3 からサーバー 4 0 のプログラムを呼び出すのではなく、サーバー 4 0 はクライアント 4 1, 4 2, 4 3 の命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供することになる。即ち、ネットワーク 4 4 を介してサーバー 4 0 がクライアントにサービスを提供する情報処理方法として、サーバー 4 0 は、プログラム（クライアントが実行するプログラム）を格納したクライアント 4 1, 4 2, 4 3 から出力されるところの前記プログラムのアドレス空間におけるアドレス情報をネットワークを介して受信し、受信したアドレス情報に対応するサービスの要求に応答するための処理を実行ユニット 5 3 P, 5 3 M で実行し、実行結果に基づくサービスをネットワーク 4 4 を介してクライアントに送信する。このように、サーバー 4 0 はクライアント 4 1, 4 2, 4 3 の命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供するというプロトコルを採用するから、サーバー 4 0 とクライアント 4 1, 4 2, 4 3 との間の通信処理が簡単である。更に、クライアント 4 1, 4 2, 4 3 において前記アドレス情報を出力する処理を規定するプログラムと、サーバ 4 0 においてアドレス情報に応じた処理を行なうプログラムとを、前述のシステム開発方法を適用して形成することができるからネットワーク 4 4 を介してクライアントにサービスを提供するデータ処理システムの低コスト化も実現できる。

【 0 1 0 1 】

なお、サーバー 4 0 の受信したアドレス情報がどのクライアントのものであるかは、特に制限されないが、前記受信アドレス情報から識別することを要しない。即ち、クライアント 4 1, 4 2, 4 3 とサーバー 4 0 の 1 対 1 の通信確立は元々システムが備えている機能を用いればよい。例えば LAN であればクライアントに割当てられる IP アドレスを用いて識別すればよいし、携帯電話網であればクライアント固有の ID 情報などを用いて識別すればよい。流用可能な識別手段を備えていない場合には前記アドレス情報にクライアント固有のプリフィックスコードを付加してサーバーに与えるようにすれば、当該プリフィックスコードによってサービスを提供すべきクライアントを識別できる。

【 0 1 0 2 】

また、図 1 0 の情報処理システムにおいて、ネットワーク 4 4 を介してプログラムを提供するような情報処理システムを想定する。即ち、インターネット等を介してプログラムのダウンロード、或いは、クライアント内で作られたプログラムにおいて使われるソフト部品をサーバーが提供するようなシステムを想定する。このとき、クライアントのシステムリソースの観点より全てのプログラムを自分で実行することが困難若しくは不経済な場合がある。これに着目して、図 1 5 に例示されるように、クライアント 4 1, 4 2, 4 3 が欲するプログラムを、少なくとも 2 つ、例えば第 1 プログラム PGM 1 と第 2 プログラム PGM 2 に分ける。サーバー 4 0 は分けられたプログラムを記憶ユニット 4 0 S に保有する。サーバー 4 0 は、一方の第 1 プログラム PGM 1 をダウンロード要求のあるクライアント 4 1, 4 2 又は 4 3 に提供し、クライアントからの要求で他方の第 2 プログラム PGM 2 を実行する。この実行は、要求を出したクライアントが必要とする結果もしくはサービスを得る為に行なわれるものである。したがって、その実行結果をクライアントに返す必要がある場合は、即座にその結果はクライアントにサービスの提供として返される。例えば第 2 のプログラムがクライアントの故障診断プログラムである場合には、診断結果に応じたサービスを提供するとよい。ソフトウェア的な故障である場合はプログラムの書換え若しくは再ダウンロードを可能にする。修理を要するハードウェア故障の場合には必要な処置若しくは

連絡先等をアナウンスする。

【 0 1 0 3 】

前記第 1 プログラム PGM 1 と前記第 2 プログラム PGM 2 は、本来一つ若しくは一体として利用される性質上、共通のソースプログラムより形成されるのが効率的である。この場合には図 7 等で説明したシステム開発方法を適用すればよく、第 1 のプログラム PGM 1 はクライアント用オブジェクトプログラム 2 0 として位置付けられるプログラムであって、第 2 のプログラム PGM 2 は対照テーブル 2 1 のサーバー用オブジェクトプログラム及びクライアントアクセス用コマンド列として位置付けられるプログラムであってよい。このとき、前記要求は、当然前記第 1 のプログラムにおけるアドレス情報を主体に行なわれることになる。

【 0 1 0 4 】

そのような処理形態は図 1 6 に示されるデータ処理方法の一つとして把握することができる。即ち、クライアント 4 1, 4 2, 4 3 がプログラムの提供をサーバー 4 0 に要求すると (S 3 0)、サーバー 4 0 は、クライアント用プログラムをネットワーク 4 4 を介して要求したクライアントに提供する (S 3 1)。例えば一体として使われるべき第 1 のプログラム PGM 1 と第 2 のプログラム PGM 2 の内、第 1 のプログラムをネットワーク 4 4 を介して要求したクライアントへ送信する。その後、サーバー 4 0 は、クライアントからのアドレス供給に応答してプログラムに関するサポートを提供する (S 3 2)。このサポートは、例えば前記第 2 のプログラム PGM 2 の実行である。サポートの内容はサーバ側のプログラムの実行だけに限らず、プログラムのバージョンアップ、破壊されたプログラムモジュールの修復等であってよい。その場合にはクライアントが保有するプログラムのバージョンを知る必要があり、前記スタック領域の情報取得と同様に所定のクライアントアクセスコマンド列を実行してバージョン情報をクライアントから取得し、そのバージョンに応じたサポートを行なえばよい。

【 0 1 0 5 】

以上本発明者によってなされた発明を実施形態に基づいて具体的に説明したが、本発明はそれに限定されるものではなく、その要旨を逸脱しない範囲において

種々変更可能であることは言うまでもない。

【0106】

例えば、C++を高級言語の一例として説明した、Fortran90、Javaなどの場合も同様に適用することができる。要するにクライアント若しくはターゲット装置にとってオーバーヘッド機能と考えられる機能をTRAP命令のようなコードとそのコードのアドレス出力等によってサーバー若しくはホスト装置に代替させればよい。また、プログラムの記憶媒体はCD-ROM、FD、ハードディスクに限定されず、MO (Magnet-Optics: 磁気光学) ディスク、フラッシュメモ리카ード等、その他の記憶形式を有する媒体であってよい。また、クライアントが出力するアドレスは、TRAP命令のようなコードのアドレスに限定されない。例えばTRAP命令のアドレスではなく、そのTRAP命令に対して数命令前に実行された命令コードのアドレスであってもよい。要するに、クライアントから供給されたアドレスに基づいて、サーバーがクライアント内の情報(代替を要求された処理を、サーバーが実現するために使う情報)へのアクセスをできるように、サーバーに代替を要求する処理を特定できるアドレスであって、クライアント内のCPUによって管理されるアドレスであればよい。

【0107】

【発明の効果】

本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。

【0108】

C++言語の例外処理機能に代表されるようなクライアント若しくはターゲット装置にとってオーバーヘッド機能と考えられる機能を、TRAP命令のような命令コードに置き換え、この命令コードの実行により例えばそのコードのアドレスを出力してサーバー若しくはホスト装置にオーバーヘッド機能の実行を代替させるから、ターゲット装置にとってオーバーヘッドになり得る高級言語機能も含めてフルセットの高級言語機能を利用可能にするシステム開発方法を実現することができる。これは、システムリソースの制限を受け易い組込みシステムにおいて、フルセットのC++言語等の高級言語を用いたシステムを開発を可能にし、

組込みシステムに関しても高級言語によるプログラムの標準化が可能になり、ワークステーション上で高級言語を用いて開発されたプログラムの移植も可能になる。

【0109】

前記システムの開発方法を実現するプログラムを記録した記憶媒体の提供によって前記システム開発方法によるシステム開発を容易化することができる。

【0110】

サービスの要求を伝送路若しくはその集合体であるネットを介して受け、そのサービスを提供するホスト装置を有するような情報処理システムを想定したとき、情報端末装置からホスト装置のプログラムを呼び出すのではなく、ホスト装置は情報端末装置の命令アドレスを受けることによりその命令アドレスに応ずるサービスを提供するというシステム構成を採用することにより、ホスト装置と情報端末装置との間の通信処理を簡素化でき、また、ネットワーク等を介して情報端末装置にサービスを提供するデータ処理システムの低コスト化、データ処理方法の効率化を実現できる。

【図面の簡単な説明】

【図1】

本発明に係るシステム開発方法で利用される各種ソフトウェアの関係を例示する説明図である。

【図2】

ターゲットマシンの開発に用いるデータ処理システムを例示するシステム構成図である。

【図3】

コンパイラによる第1の記述に関するコンパイル機能の詳細を例示する説明図である。

【図4】

C++の言語機能を模式的に示した説明図である。

【図5】

C++言語におけるオブジェクトの解放処理を実現するためにスタックフレー

ムが通常備えるフレームポインタ及び解放処理関数アドレスの領域を示す説明図である。

【図 6】

フレームポインタ及び解放処理関数アドレスの領域を省いたスタックフレームの説明図である。

【図 7】

クライアントのシステムを開発するときのシステム開発方法を例示するフローチャートである。

【図 8】

図 7 の方法で生成されるプログラムを例示する説明図である。

【図 9】

クライアントのシステムを開発するときのシステム開発装置を例示するシステム構成図である。

【図 1 0】

ネットワークを介してサーバーがクライアントにサービスを提供する情報処理システムを例示するブロック図である。

【図 1 1】

クライアントの具体例を示すブロック図である。

【図 1 2】

サーバーの具体例を示すブロック図である。

【図 1 3】

クライアントによるクライアント用オブジェクトプログラムの実行制御状態を例示するフローチャートである。

【図 1 4】

サーバによるサーバ用オブジェクトプログラム及びクライアントアクセス用コマンド列の実行制御状態を例示するフローチャートである。

【図 1 5】

図 1 0 の情報処理システムにおいてネットワークを介してプログラムを提供するような情報処理システムを想定したときのサーバーの機能を概略的に示したブ

ロック図である。

【図 1 6】

ネットワーク上でプログラムのアドレス情報を用いてサービスを提供するデータ処理方法の一例を示すフローチャートである。

【符号の説明】

- 1 ホストマシン
- 3 ターゲットマシン
- 4 ソースプログラム
- 5 コンパイラ
- 6 デバッガスクリプト
- 7 ロードモジュール
- 8 アプリケーションプログラム
- 9 デバッグも似たプログラム
- 10 デバッガ
- 11 通信機能プログラム
- P1g 関数 g のフレームポインタ
- P2g 関数 g の解放処理関数のアドレス
- 20 ソースプログラム
- 20 クライアント用オブジェクトプログラム
- 21 サーバ用オブジェクトプログラム及びクライアントアクセスコマンド列

(対照テーブル)

- 30 システム開発装置
- 31, 32, 34 記憶媒体
- 34 公衆回線
- 40 サーバー
- 40S 記憶ユニット
- 41, 42, 43 クライアント
- 41M, 42M, 43M メモリ
- 44 ネットワーク

5 0 C P U

5 3 計 算 機

5 3 P プ ロ セ ッ サ

5 3 M R A M

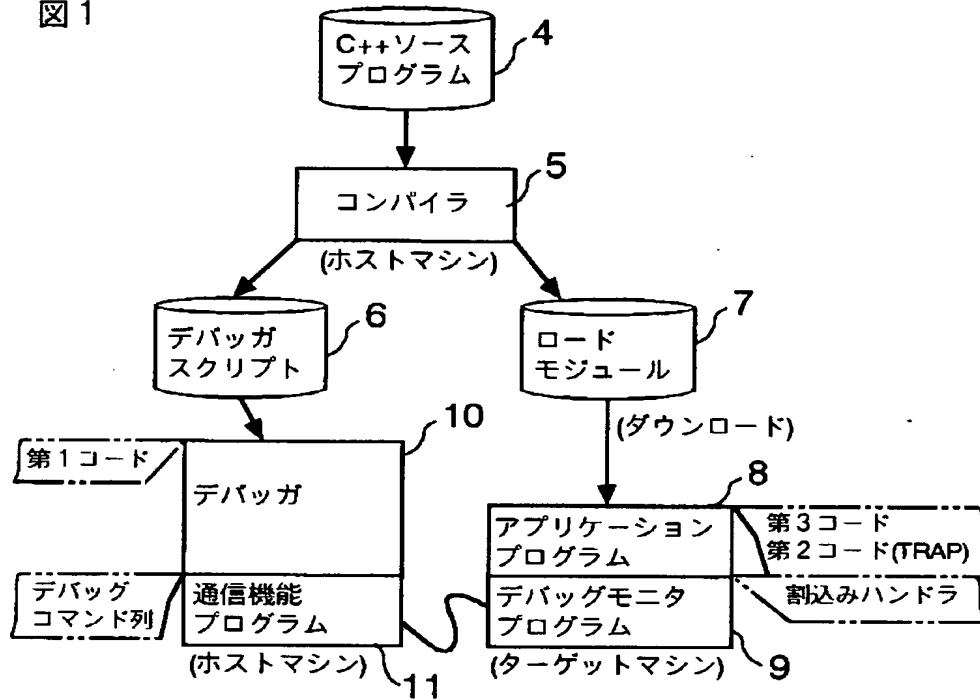
P G M 1 第 1 の プ ロ グ ラ ム

P G M 2 第 2 の プ ロ グ ラ ム

【書類名】 図面

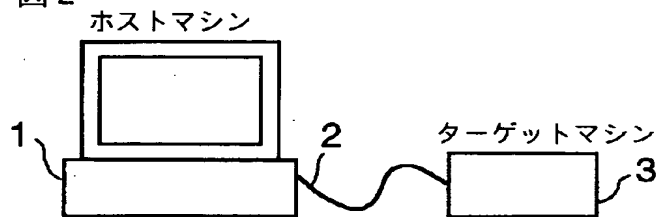
【図 1】

図 1



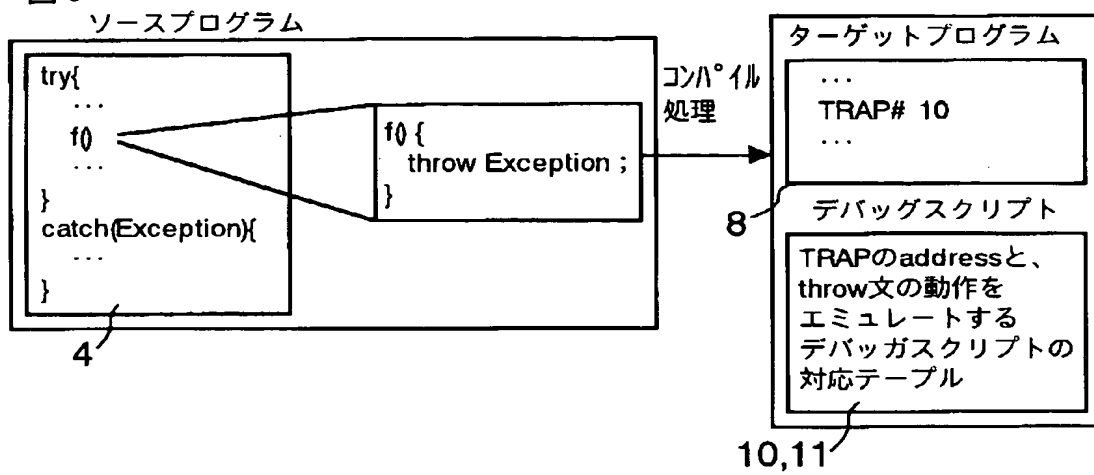
【図 2】

図 2



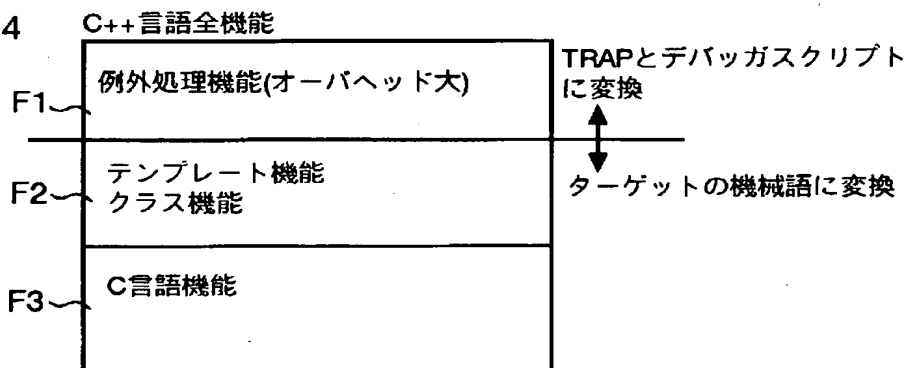
【図 3】

図 3



【図 4】

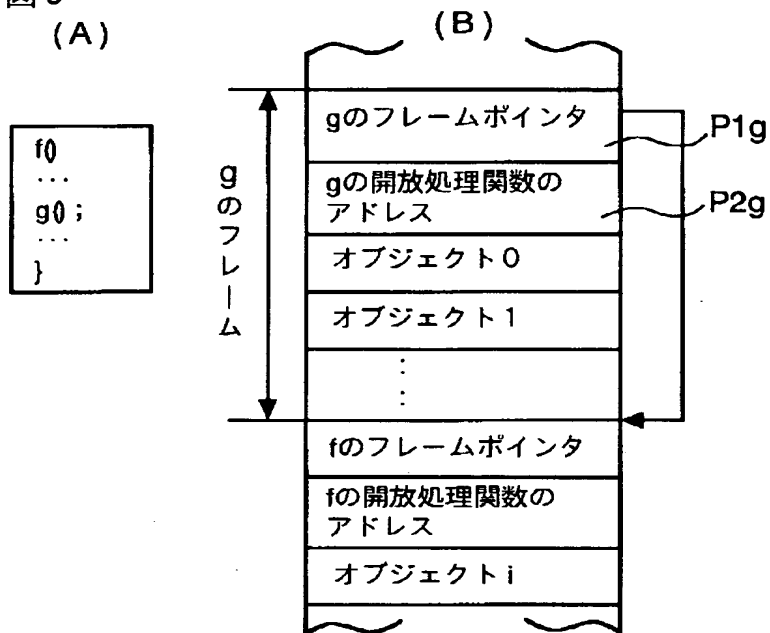
図 4



【図 5】

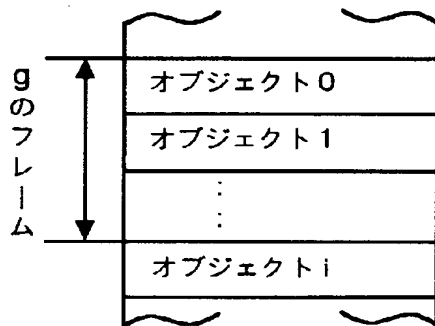
図 5

(A)



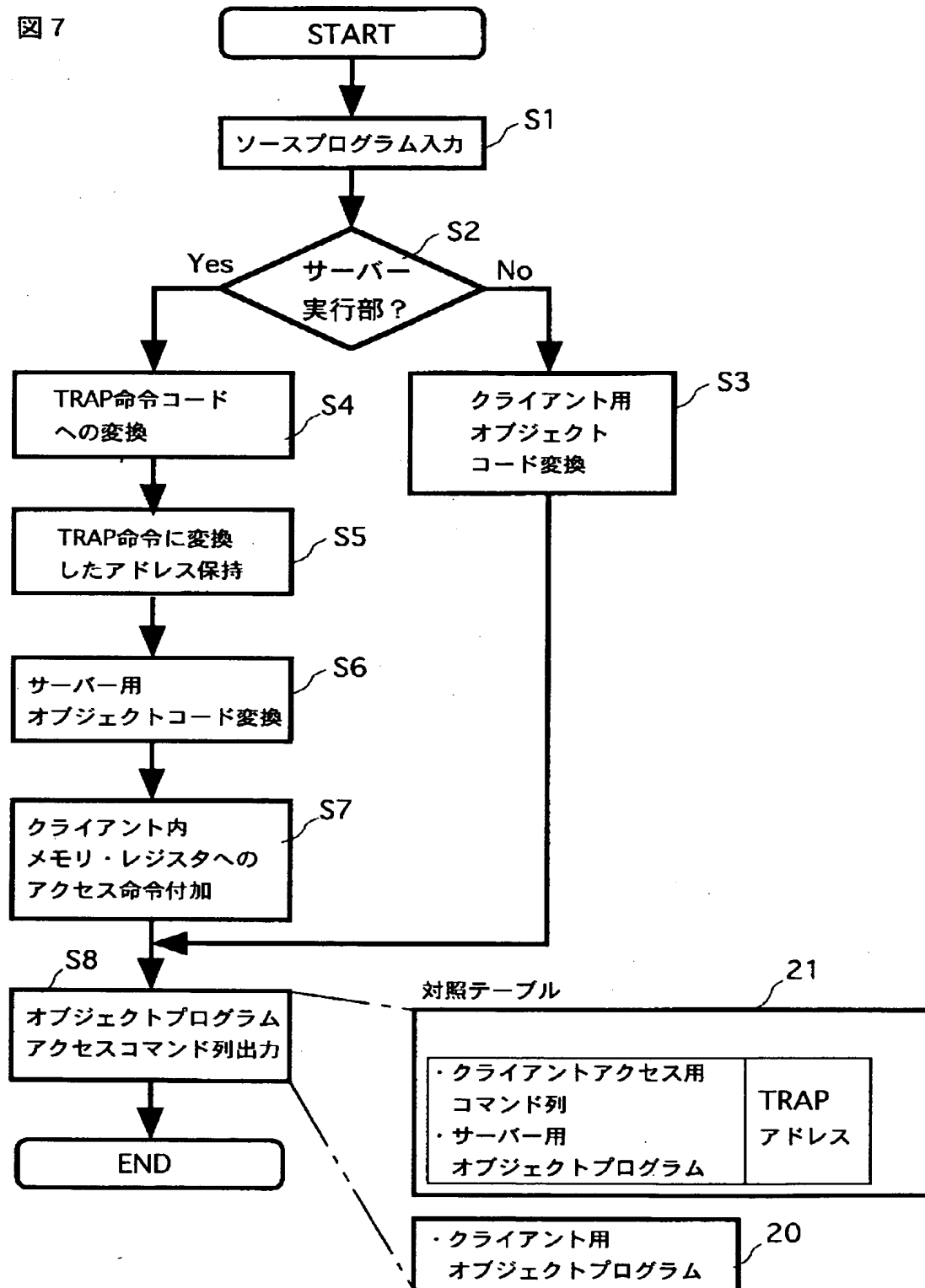
【図 6】

図 6



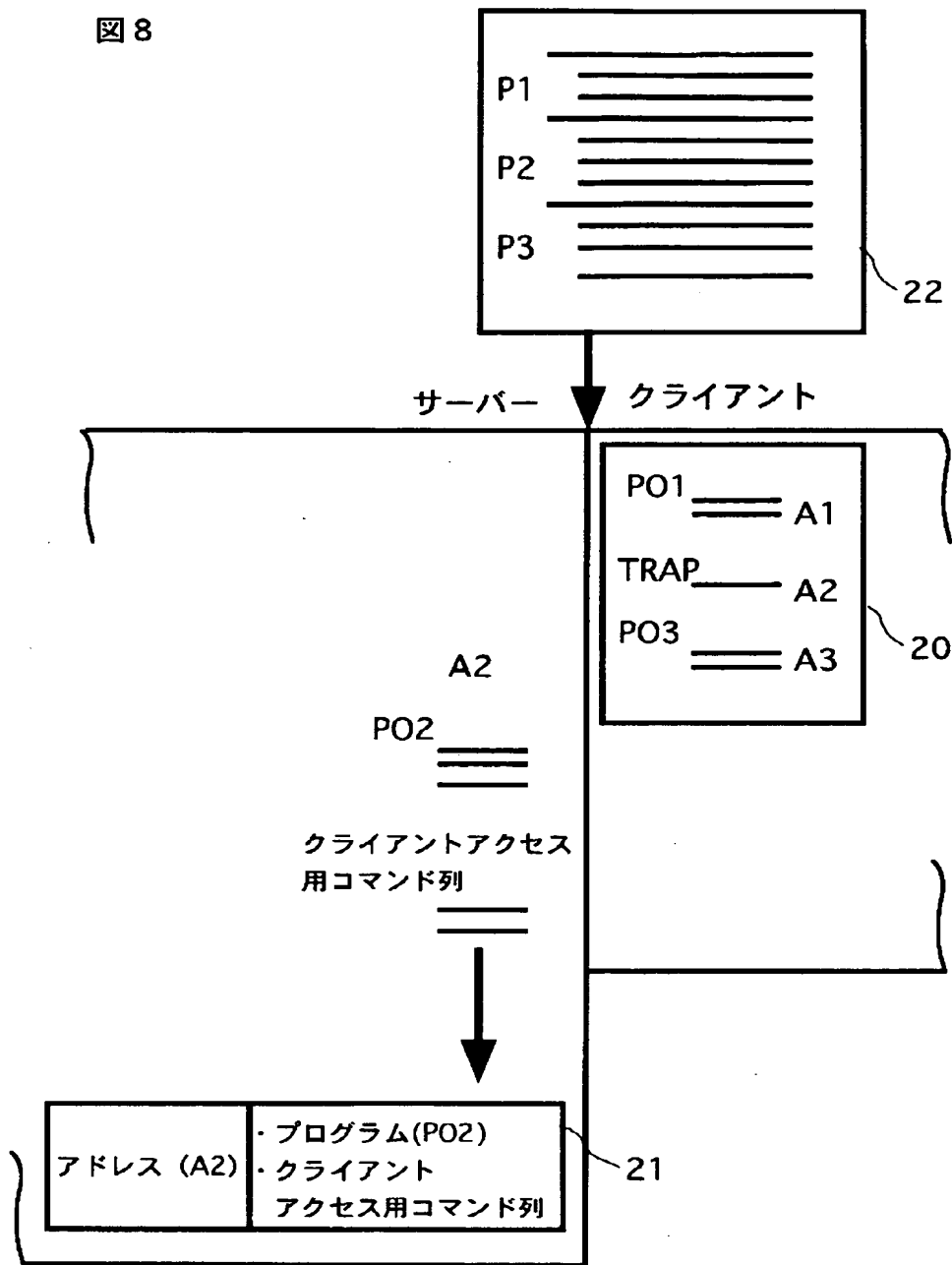
【図 7】

図 7



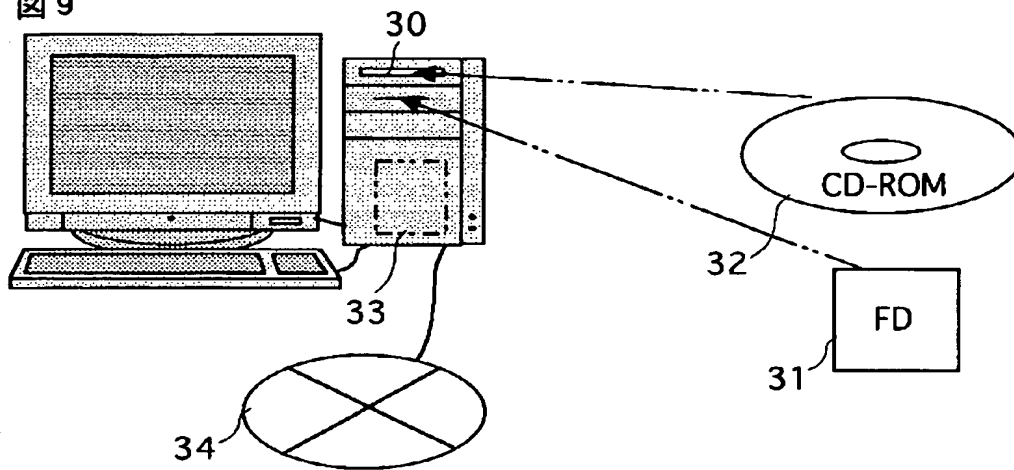
【図 8】

図 8



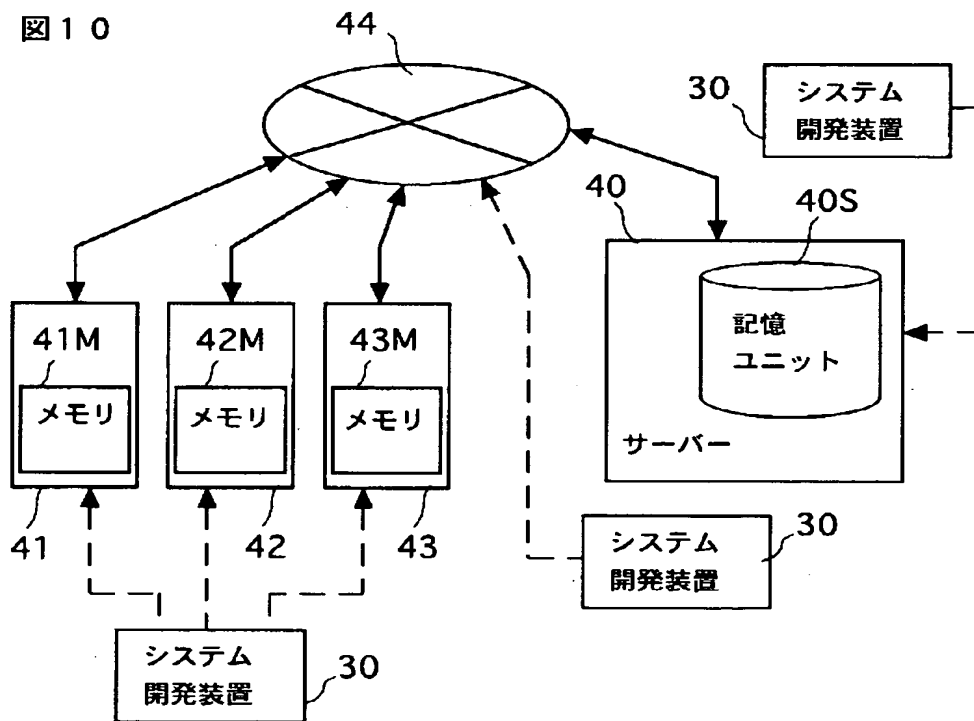
【図 9】

図 9



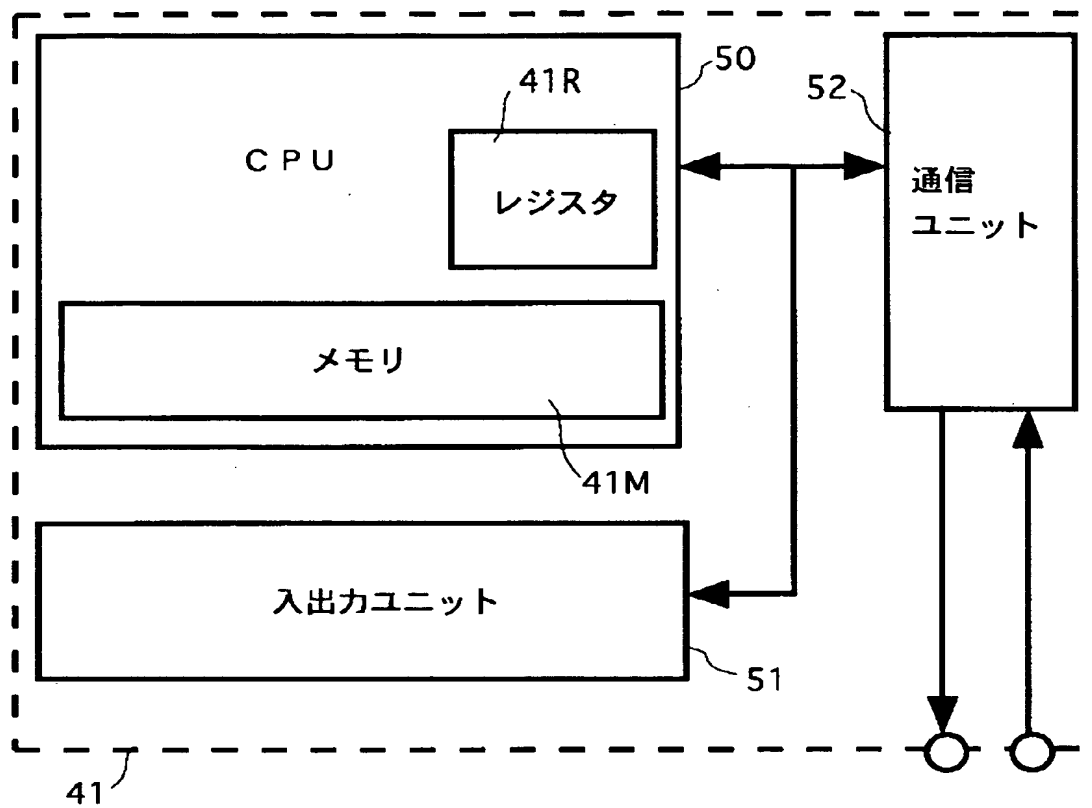
【図 10】

図 10



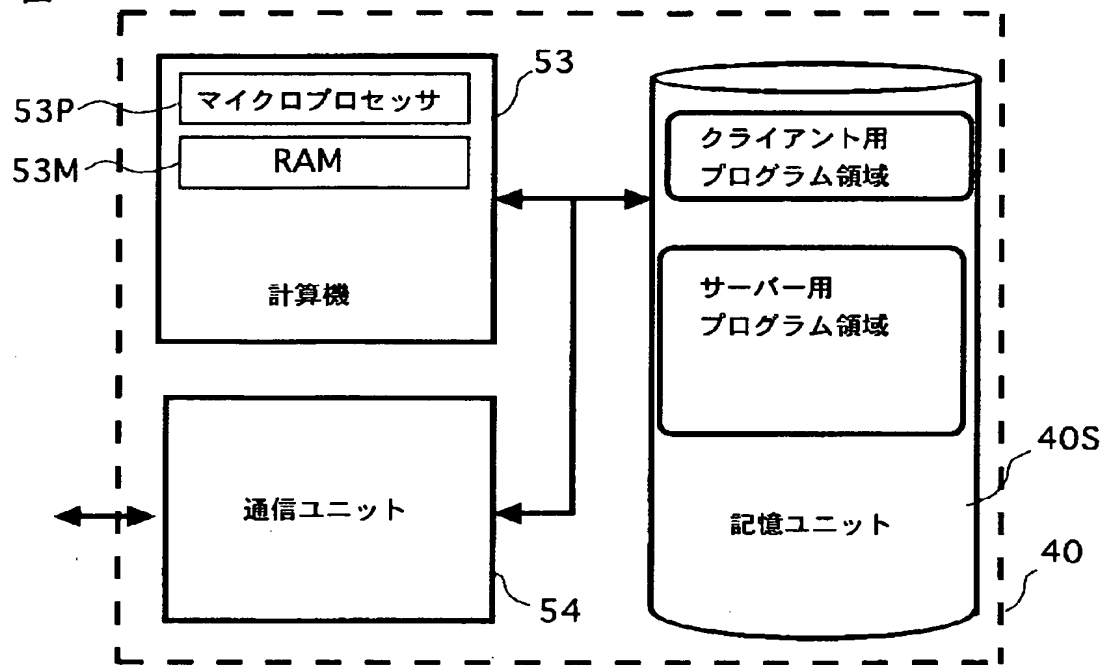
【図 1 1】

図 1 1



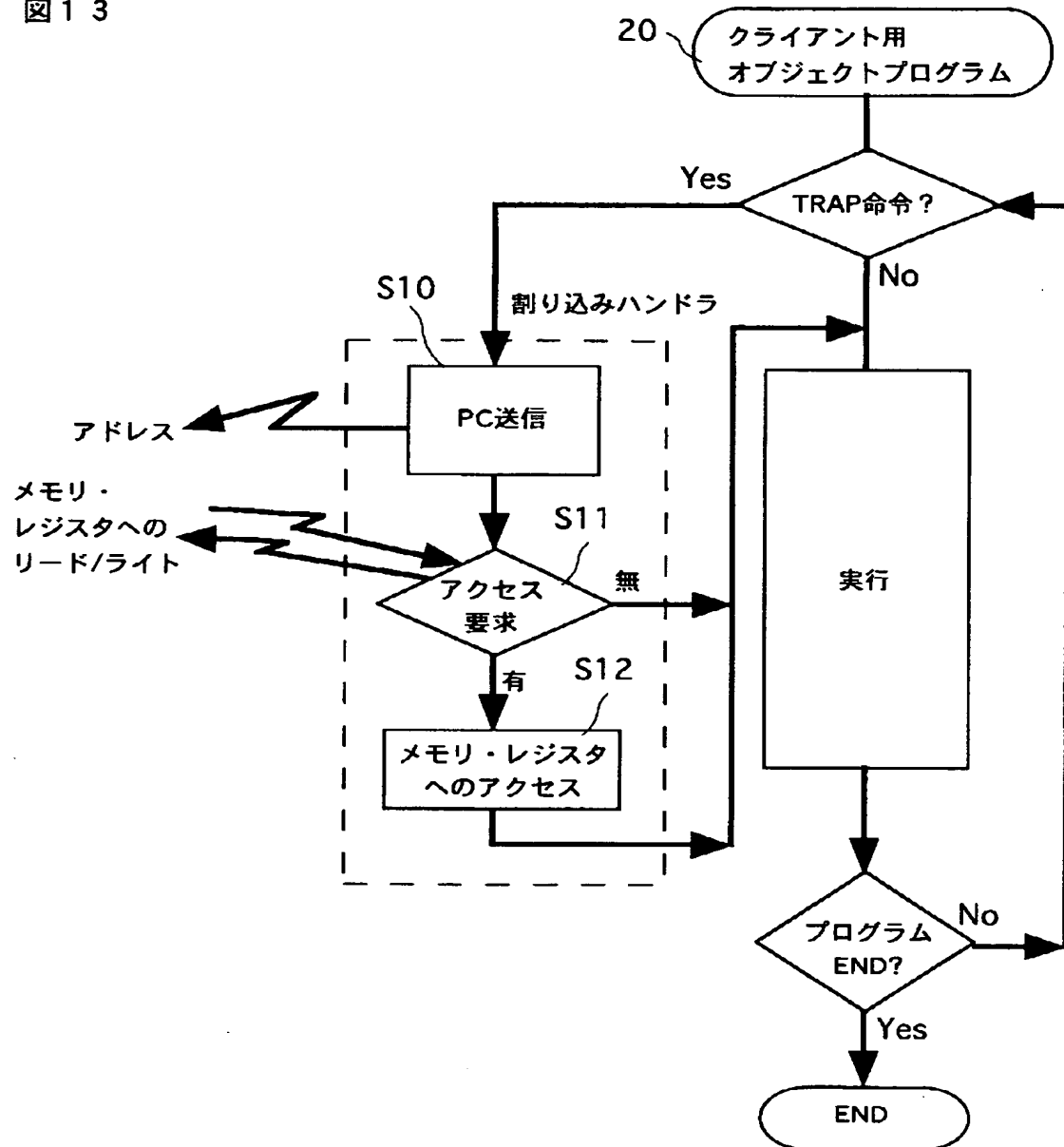
【図 12】

図 12



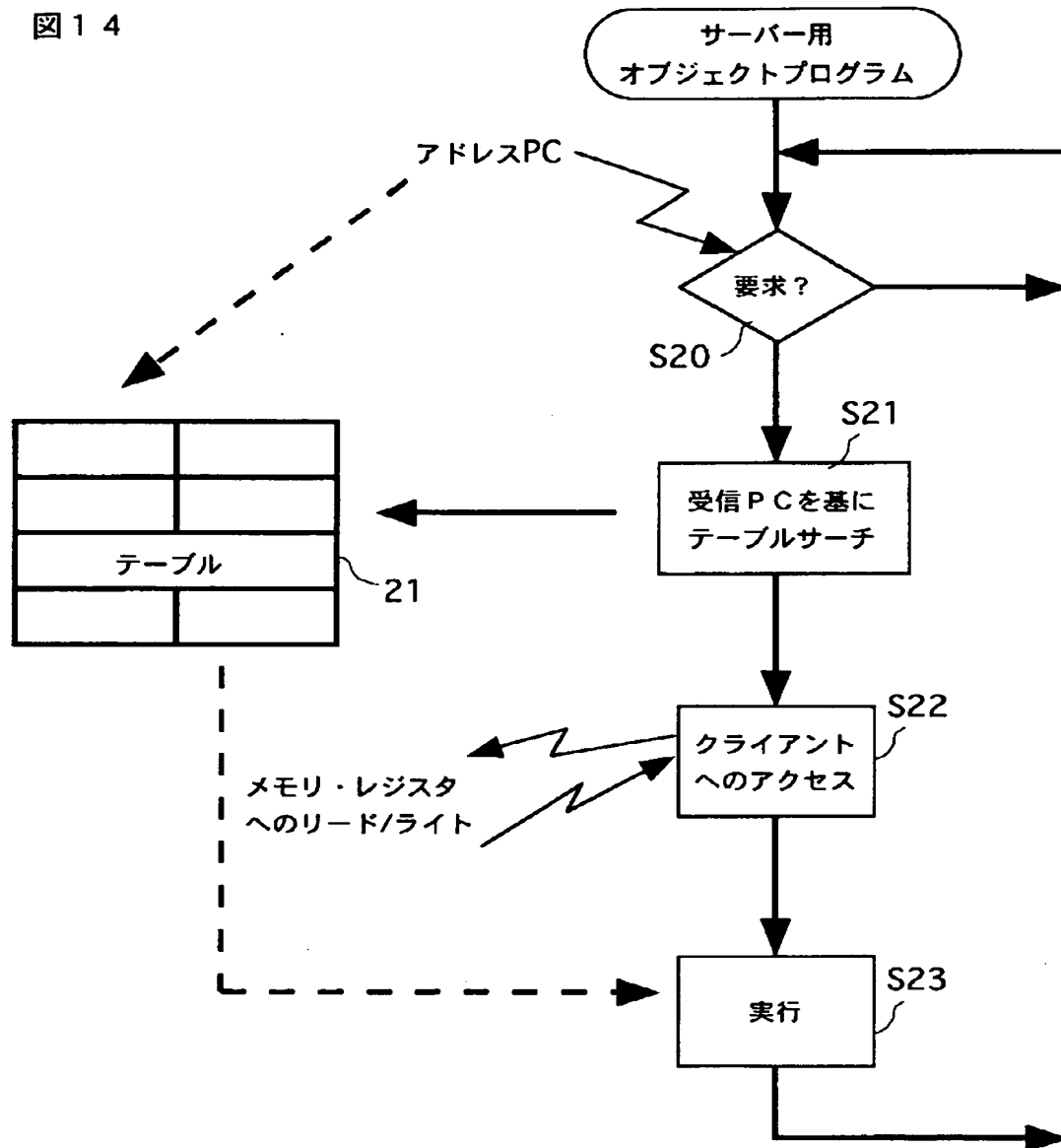
【図13】

図13



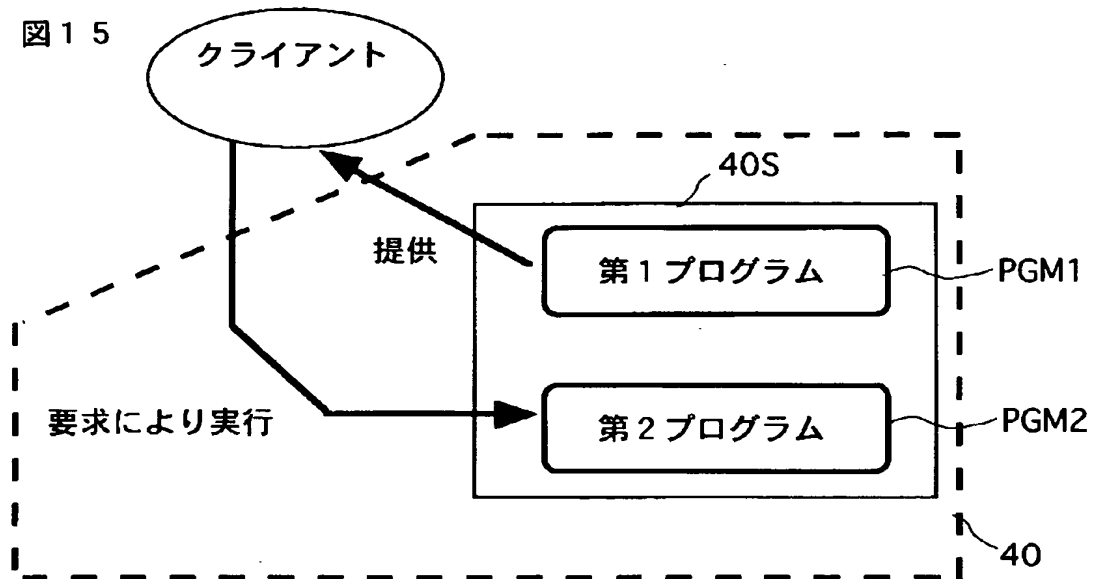
【図 1 4】

図 1 4



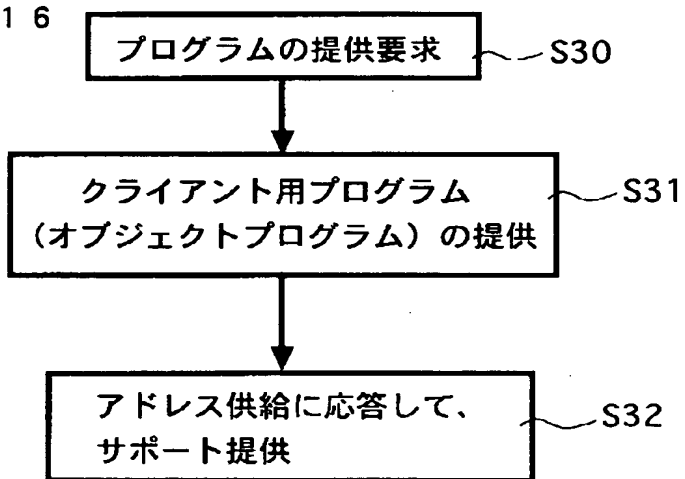
【図 1 5】

図 1 5



【図 1 6】

図 1 6



【書類名】 要約書

【要約】

【課題】 ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化、及びクライアントとの間の通信の簡素化を実現する。

【解決手段】 サービスの要求を伝送路若しくはネット（４４）を介して受け、そのサービスを提供するサーバー（４０）を有するような情報処理システムを想定したとき、クライアント（４１，４２，４３）からサーバーのプログラムを呼び出すのではなく、サーバーはクライアントからそのアドレス空間の命令アドレスのようなアドレス情報を受けることによりそのアドレス情報に応ずるサービスを提供する。このプロトコルを採用するから、サーバーとクライアントとの間の通信処理が簡素であり、ネットワーク等を介してクライアント若しくは情報端末装置にサービスを提供するデータ処理システムの低コスト化も実現できる。

【選択図】 図１０

出 願 人 履 歴 情 報

識別番号 [000005108]

1. 変更年月日 1990年 8月31日
[変更理由] 新規登録
住 所 東京都千代田区神田駿河台4丁目6番地
氏 名 株式会社日立製作所